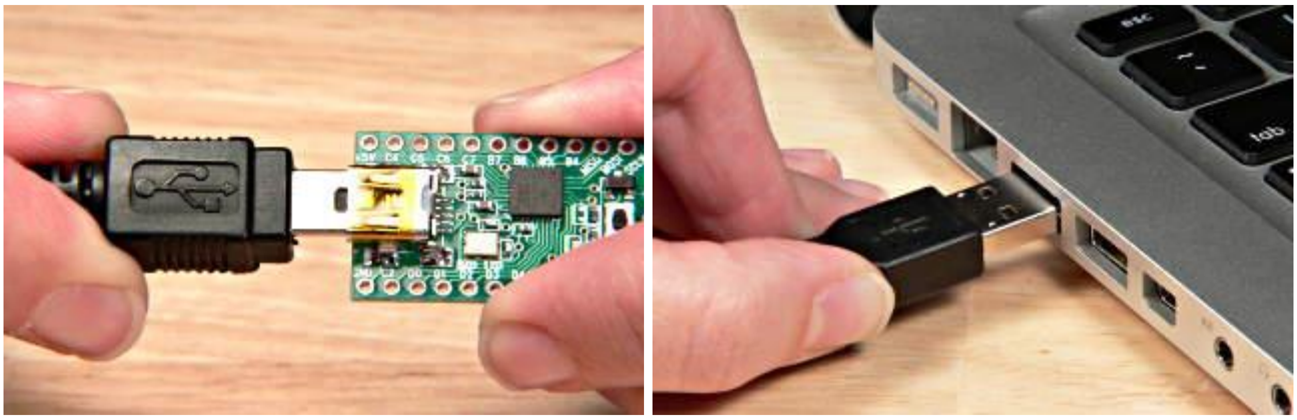
Teensy Getting Started Guide

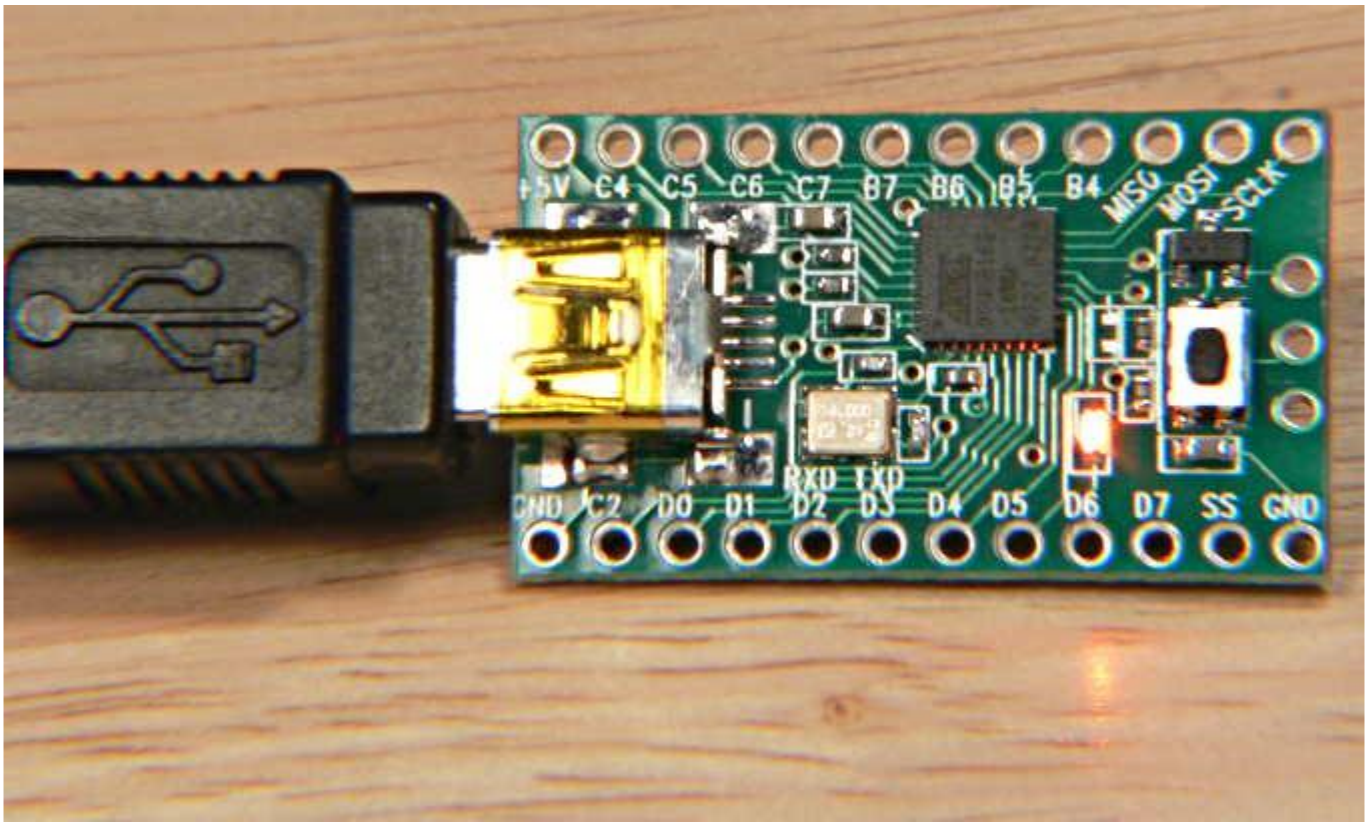
Step #1: Teensy First Use

The first step is to plug your new Teensy in using the USB cable.



Simply plug the Teensy board in to a USB port to get started.

All brand new Teensy boards come with the LED blink program pre-loaded. You should see the Orange LED blink slowly, 1 second on, 1 second off.



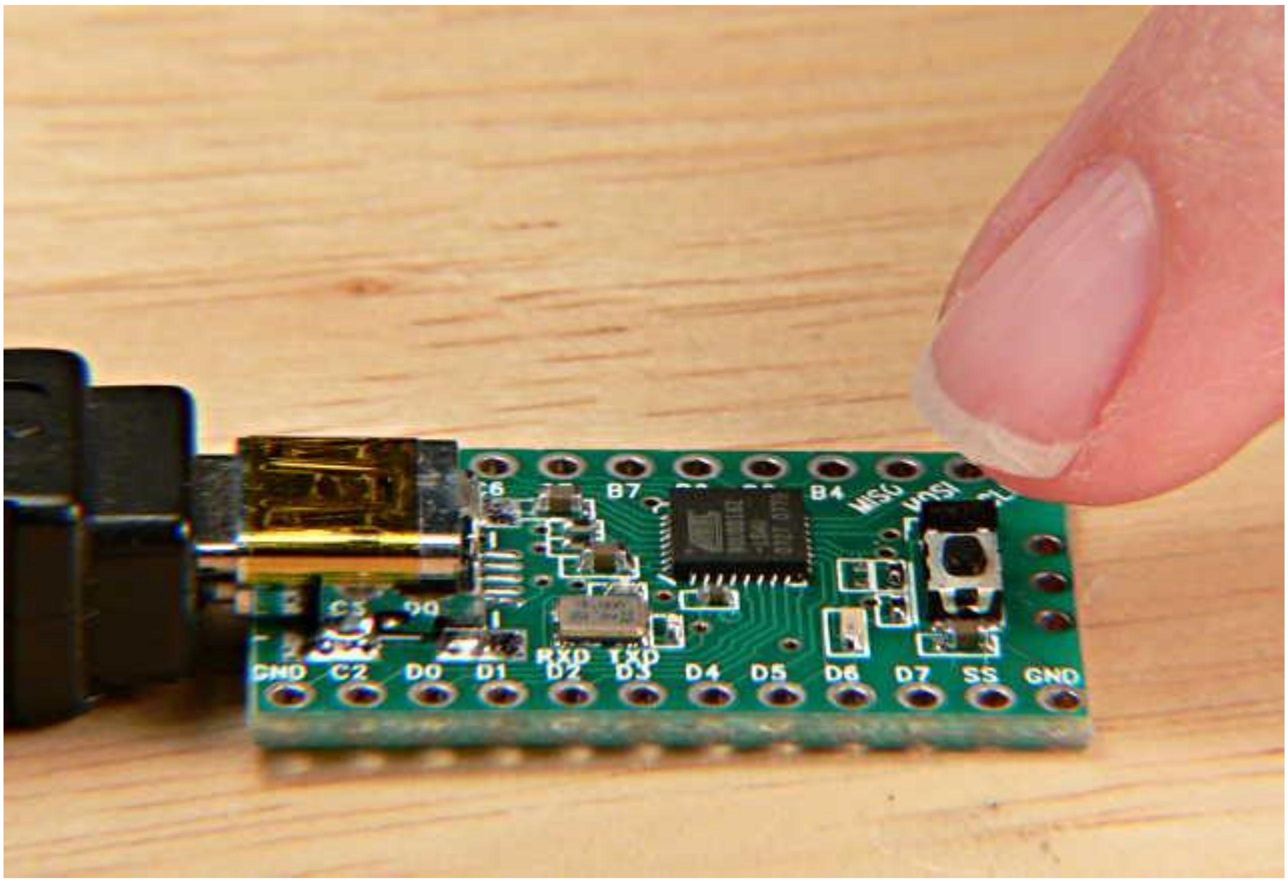
LED blink program is pre-programmed on all new Teensy boards

If your Teensy is not brand new, it will run whatever program the previous owner loaded, perhaps causing your computer to recognize new hardware and require drivers. Just cancel any driver setup. In the next step you will load the LED blink program.

If your Teensy should blink but does not, please check the USB cable to make sure it is properly connected. Your computer must be on so it provides power to the USB port. If you still experience problems, try a different USB port or different computer.

HalfKay Bootloader Mode

Your Teensy contains 2 programs, the LED blink (user program) and the HalfKay bootloader. HalfKay, together with the Teensy Loader software, allows you to easily load new programs. Normally, the user program will run. To run HalfKay, press and release the tiny pushbutton.



Press and release the pushbutton to enter HalfKay bootloader mode.

The LED blinking will stop, because HalfKay never lights the LED.

Your computer may briefly display a message or beep to indicate new hardware. Windows, MacOS-X and Linux have built-in drivers that are automatically used, so you never need to install drivers for HalfKay!

Understanding When Each Program Runs

User Program Runs When...

- Power is applied
- Reboot button clicked in Teensy Loader
- (In Auto mode) Immediately after download

HalfKay Runs When...

- Pushbutton is pressed (and released)
- If user program jumps to Teensy code

Nothing Runs When...

- Pushbutton held down

Loading New Programs

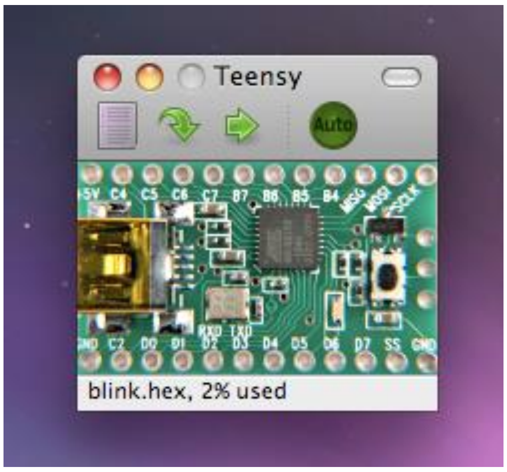
The next step is to install and use the Teensy Loader application to download different programs to your Teensy board.

Step #2: The Teensy Loader Application

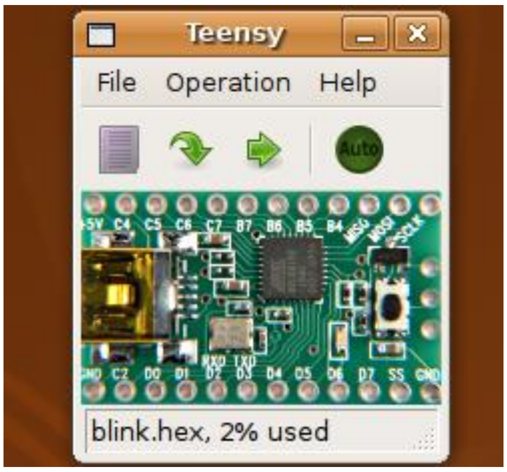
The Teensy Loader program communicates with your Teensy board when the HalfKay bootloader is running, so you can download new programs and run them.

Update: Teensy 3.0 is supported by Teensyduino. After installation, Teensy Loader is in hardware/tools. It is run automatically when using Verify or Upload within the Arduino software.

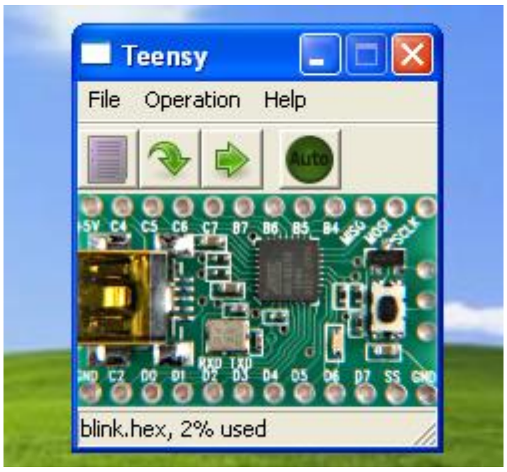
Teensy Loader is available for these operating systems:



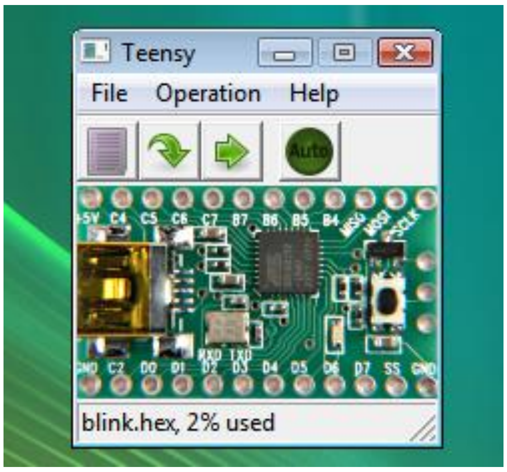
Macintosh OS X 10.5



Linux (Ubuntu)



Windows XP



Windows 7 & Vista

Command Line Version

The Teensy Loader is also available in a command line version for advanced users. It works on Mac, Windows, Linux and BSD Unix.

Revision History

The Teensy Loader version is shown in the About window, available from the Help menu, or "Teensy" menu on Mac OS X.

1.15

- Version number is now synchronized to Teensyduino
- Improve .elf chip detection for Teensy 3.0

1.07

- Support for Teensy 3.0

1.06

- Fix off-screen issues with Windows multiple monitor desktops
- Support for Teensyduino 0.9

1.05

- Support Teensy 2.0 board
- Improve .elf chip detection
- Program EEPROM, if found in .elf file

1.04

- Do not program when .elf file is clearly compiled for the wrong chip
- Display Teensy++ image when communicating with a Teensy++ board
- More careful program close, hopefully fixes crash-on-quit on Mac OS X

1.03

- Wait for reboot, (hopefully) solves incorrect auto mode disable, especially on Windows
- More status reported to Teensyduino (to be used by version 0.4)
- Fix memory leaks on Mac OS X

1.02

- More remote control features (for Teensyduino 0.2)

- Never show "stale lock" message at startup on Mac OS X and Linux
- Add support for AT90USB646 chip

1.01

- Add remote control support (for Teensyduino)
- Add universal binary for Mac OS X
- Fix drag and drop on Mac OS X
- Fix filename disappearing on Linux and Windows
- USB HID messages now appear in verbose info window

1.00

- Initial Release

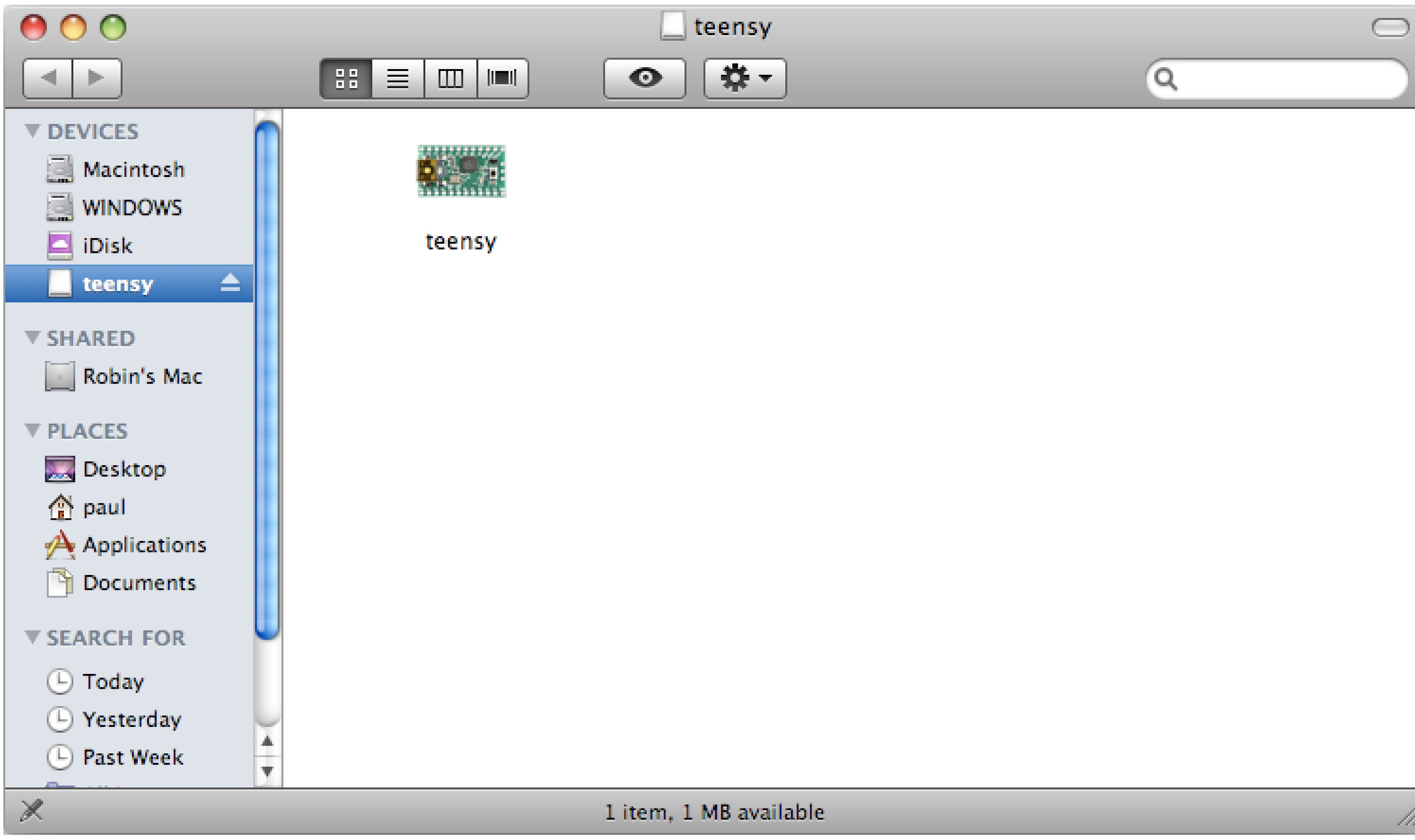
Using The Teensy Loader on Mac OS X

Update: Teensy 3.0 is supported by Teensyduino. After installation, Teensy Loader is in hardware/tools. It is run automatically when using Verify or Upload within the Arduino software. This version on this web page has not been updated for Teensy 3.0. Please install Teensyduino to use Teensy 3.0.

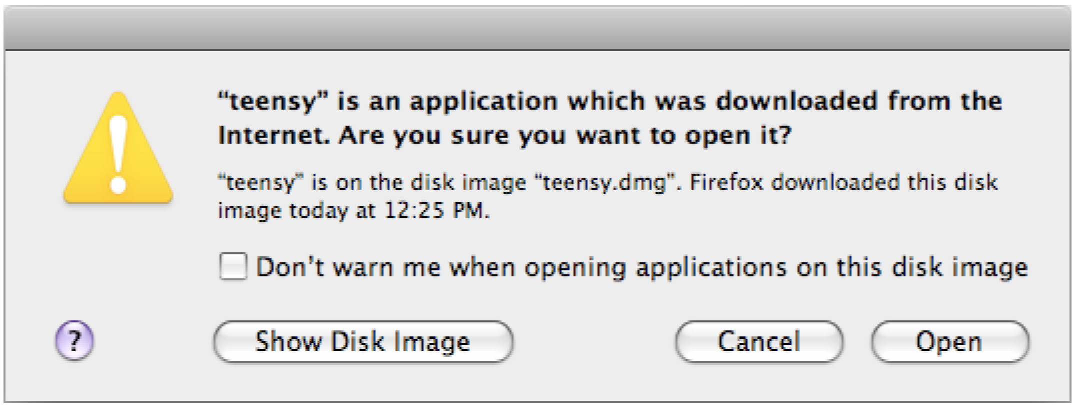
Files to download:

- Teensy Loader Disk Image (3.5 megabytes, Mac OS X 10.5 and later)
- LED Blink, Both Slow & Fast

When you open the disk image, it contains 1 item, the Teensy Loader application.



You may copy the Teensy Loader to your hard drive, perhaps even add it to your dock. But it will run perfectly directly from the disk image, so just double click it.

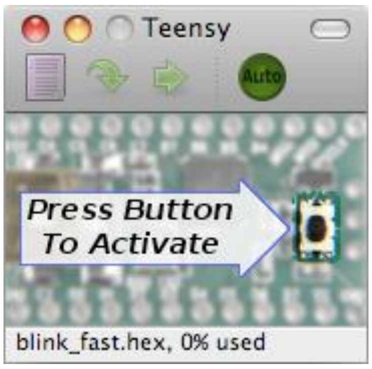

"teensy" is an application which was downloaded from the Internet. Are you sure you want to open it?

"teensy" is on the disk image "teensy.dmg". Firefox downloaded this disk image today at 12:25 PM.

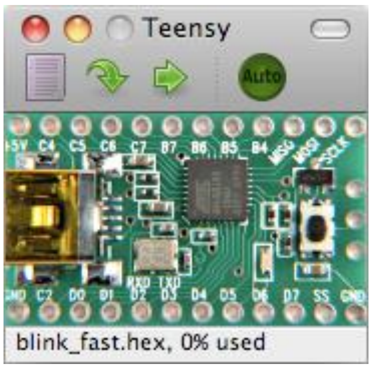
☐ Don't warn me when opening applications on this disk image

Yes, the Teensy Loader is safe, so click Open.

The Teensy Loader should appear as a small window. If the Teensy board is running the LED blink, or not connected, you should see this window.



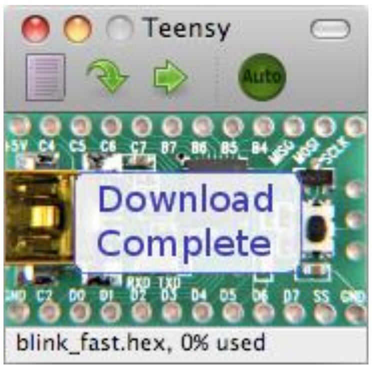
If HalfKay is running, you should see this. If not, simply make sure your Teensy board is connected, and press the pushbutton to run HalfKay. The Teensy Loader will quickly recognize it.


Teensy
Auto
+5V C4 C5 C6 C7 B7 B6 B5 B4 MISO MOSI SCLK
GND C2 D0 D1 RXD D2 TXD D3 D4 D5 D6 D7 SS GND

blink_fast.hex, 0% used

From the File name, choose "Open HEX File" and open blink_fast.hex, which you downloaded using the link at the top of this page. You can also open the file using the toolbar button, or using drag-and-drop onto the upper portion of the window. When the file is opened, the filename and percentage usage of the Teensy's memory are shown.

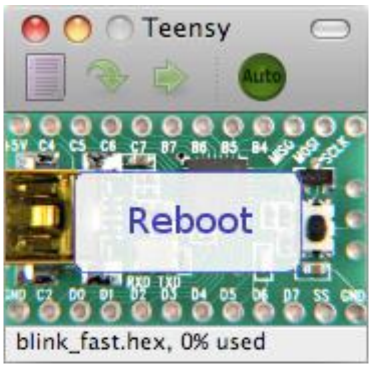
Select "Program" from the "Operations" menu, or click the Program button on the tool bar. The download should be very quick, perhaps too fast to see a tiny progress bar appear, but you should see the "Download Complete" message.



Download Complete

blink_fast.hex, 0% used

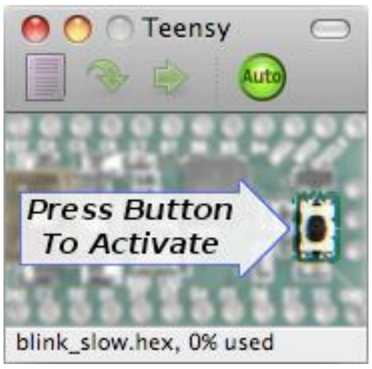
Choose "Reboot" from the "Operations" menu, or click the Reboot button on the tool bar. Your Teensy board should immediately begin running the fast LED blink program!



Reboot

blink_fast.hex, 0% used

To try Automatic mode, simply click the "Auto" button, or choose "Automatic Mode" from the Operations menu. The Auto button should illuminate bright green. Then, open the slow LED blink HEX file.



When you press the button on your Teensy board, Automatic Mode will quickly program and reboot your board. Normally, while working on a project, you can leave the Teensy Loader in Automatic Mode. Teensy Loader always reads the latest version of your HEX file, so you can just compile your code, then press the pushbutton to program and run your code.

Mac OS 10.4 Tiger (does not work)

Apple changed their HID manager API in 10.5 Leopard. Apple Technical Note 2187 has all the details. Teensy Loader uses this API and the IOHIDDeviceSetReport function to write your code.

Apple's earlier API does not seem to have any functions that can write to HID reports. All functions only read. If you know of any way to set HID reports in Tiger's HID API, please email paul@pjrc.com

Using The Teensy Loader on Ubuntu Linux

Update: Teensy 3.0 is supported by Teensyduino. After installation, Teensy Loader is in hardware/tools. It is run automatically when using Verify or Upload within the Arduino software. This version on this web page has not been updated for Teensy 3.0. Please insta ll Teensyduino to use Teensy 3.0.

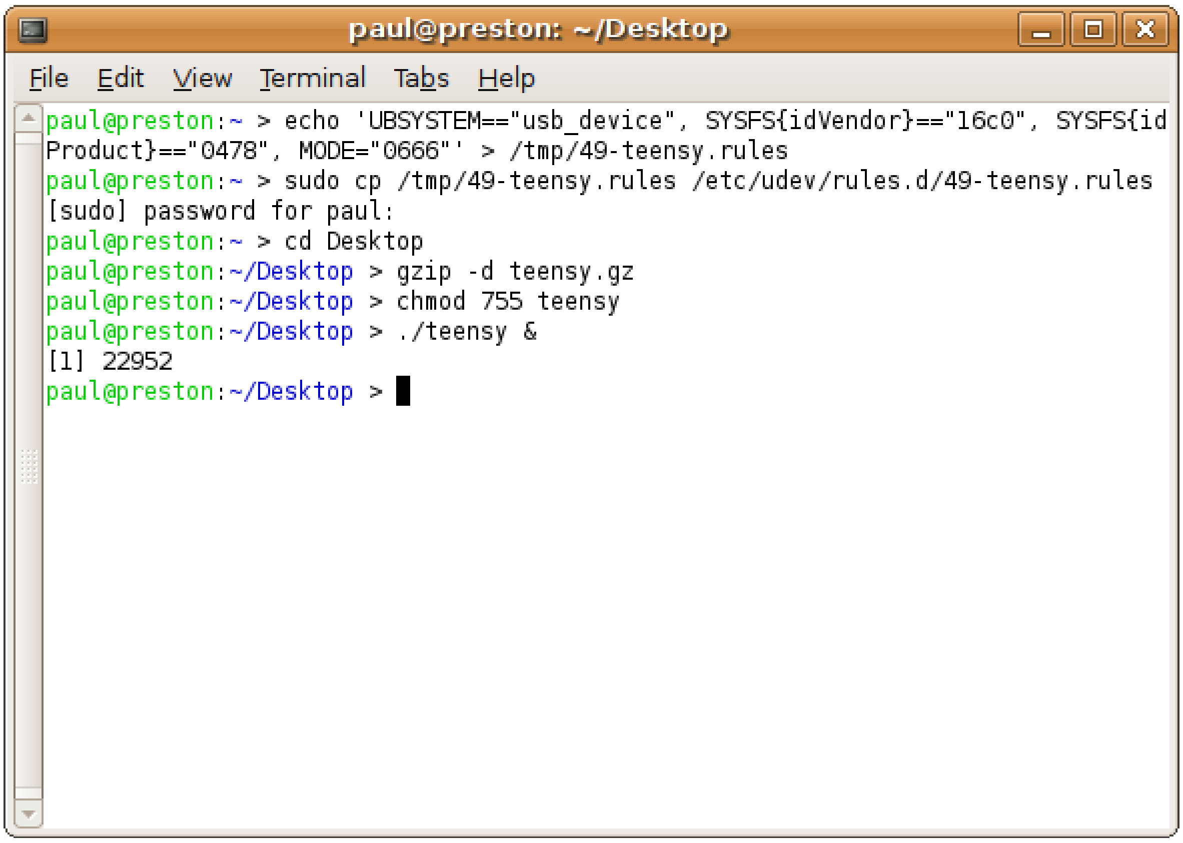
Files to download:

- Download Teensy Program (32 bit)
- Download Teensy Program (64 bit)
- Download Teensy Program+Utils (Raspberry Pi - Experimental)
- LED Blink, Both Slow & Fast
- Linux udev rules

Ubuntu and other modern Linux distibutions use udev to manage device files when USB devices are added and removed. By default, udev will create a device with read-only permission which will not allow to you download code. You must place the udev rules file at /etc/udev/rules.d/49-teensy.rules

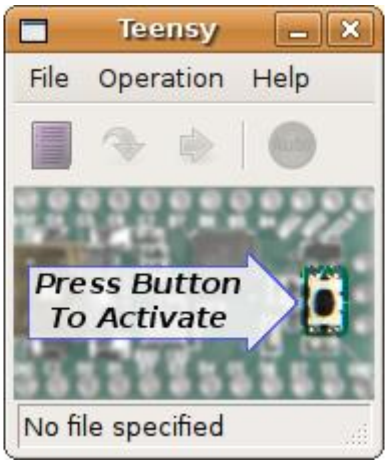
The file is gzip compressed, so you must extract it and then give it execute permission, and then you may run it. Here are sample commands. Please adjust as necessary, for example the download directory.

```
cd Desktop
sudo cp 49-teensy.rules /etc/udev/rules.d/
gzip -d teensy.gz
chmod 755 teensy
./teensy &
```

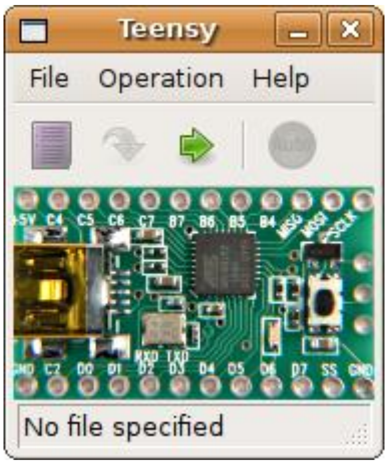


```
paul@preston:~ > echo 'UBSYSTEM=="usb_device", SYSFS{idVendor}=="16c0", SYSFS{id
Product}=="0478", MODE="0666"' > /tmp/49-teensy.rules
paul@preston:~ > sudo cp /tmp/49-teensy.rules /etc/udev/rules.d/49-teensy.rules
[sudo] password for paul:
paul@preston:~ > cd Desktop
paul@preston:~/Desktop > gzip -d teensy.gz
paul@preston:~/Desktop > chmod 755 teensy
paul@preston:~/Desktop > ./teensy &
[1] 22952
paul@preston:~/Desktop >
```

The Teensy Loader should appear as a small window. If the Teensy board is running the LED blink, or not connected, you should see this window.

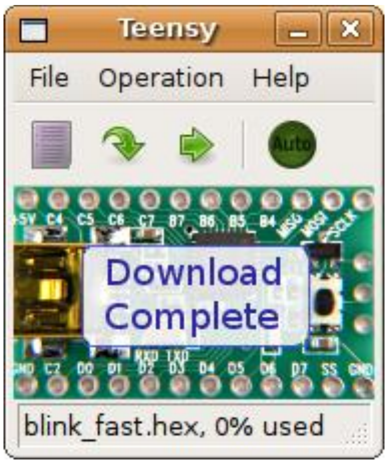


If HalfKay is running, you should see this. If not, simply make sure your Teensy board is connected, and press the pushbutton to run HalfKay. The Teensy Loader will quickly recognize it.

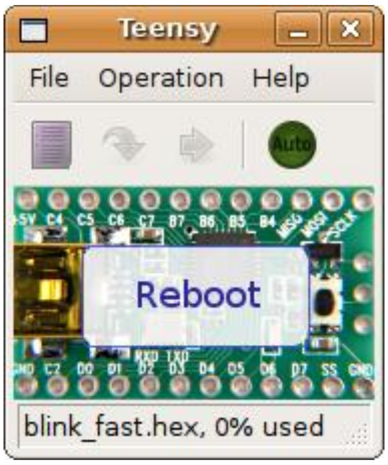


From the File name, choose "Open HEX File" and open blink_fast.hex, which you downloaded using the link at the top of this page. You can also open the file using the toolbar button, or using drag-and-drop onto the upper portion of the window. When the file is opened, the filename and percentage usage of the Teensy's memory are shown.

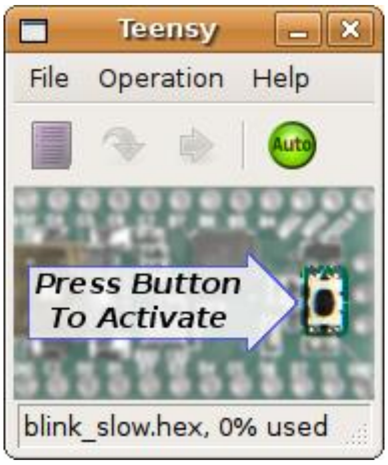
Select "Program" from the "Operations" menu, or click the Program button on the tool bar. The download should be very quick, perhaps too fast to see a tiny progress bar appear, but you should see the "Download Complete" message.



Choose "Reboot" from the "Operations" menu, or click the Reboot button on the tool bar. Your Teensy board should immediately begin running the fast LED blink program!



To try Automatic mode, simply click the "Auto" button, or choose "Automatic Mode" from the Operations menu. The Auto button should illuminate bright green. Then, open the slow LED blink HEX file.



When you press the button on your Teensy board, Automatic Mode will quickly program and reboot your board. Normally, while working on a project, you can leave the Teensy Loader in Automatic Mode. Teensy Loader always reads the latest version of your HEX file, so you can just compile your code, then press the pushbutton to program and run your code.

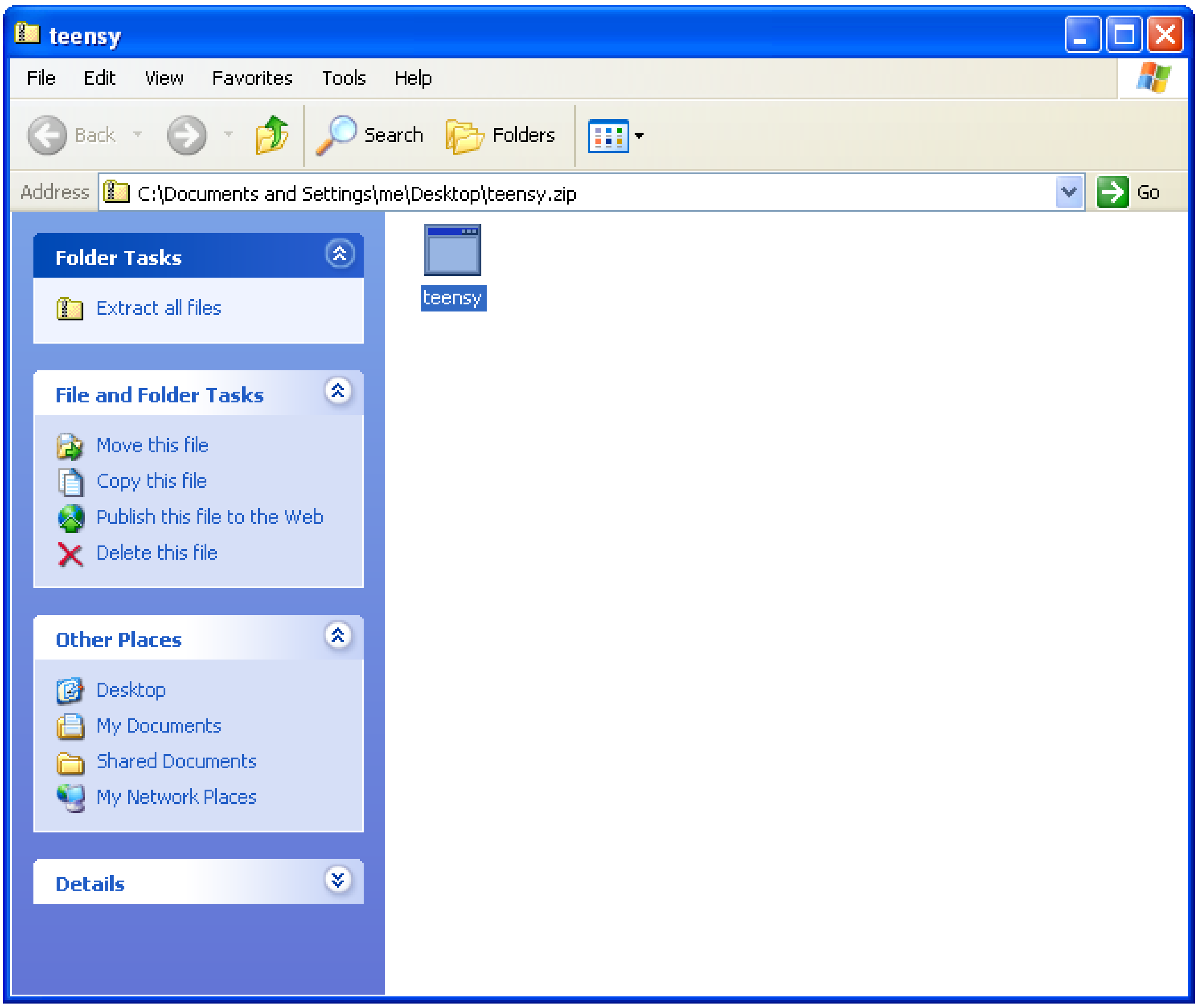
Using The Teensy Loader on Windows XP

Update: Teensy 3.0 is supported by Teensyduino. After installation, Teensy Loader is in hardware/tools. It is run automatically when using Verify or Upload within the Arduino software. This version on this web page has not been updated for Teensy 3.0. Please insta ll Teensyduino to use Teensy 3.0.

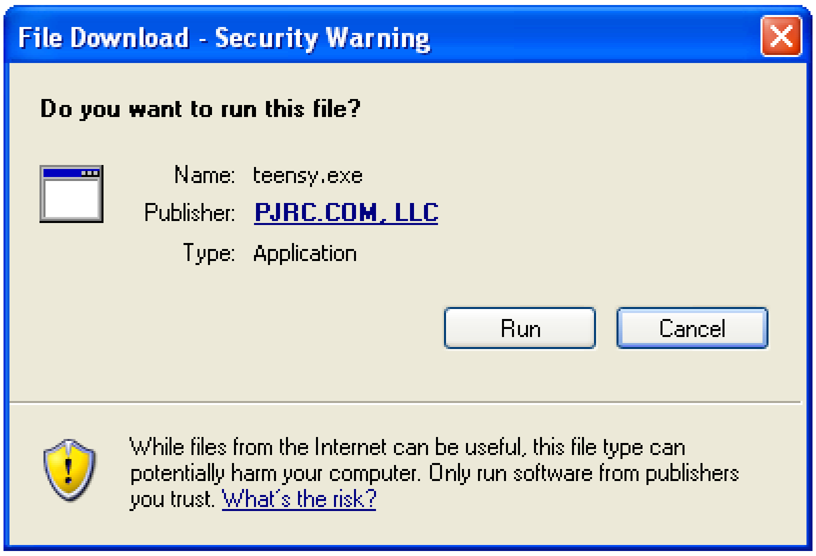
Files to download:

- Teensy Loader Program
- LED Blink, Both Slow & Fast

TEENSY.EXE is a single file application, so there is no installer required. You may copy it anywhere on your computer.



Simply double click the application to run it. The first time it runs, Microsoft will ask you to confirm. This window should show "PJRC.COM, LLC" as the publisher.

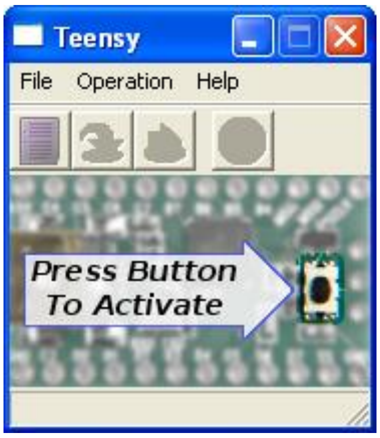

File Download - Security Warning

Do you want to run this file?

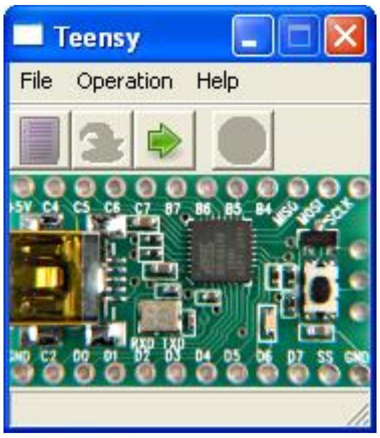
Name: teensy.exe
Publisher: PJRC.COM, LLC
Type: Application

While files from the Internet can be useful, this file type can potentially harm your computer. Only run software from publishers you trust. What's the risk?

The Teensy Loader should appear as a small window. If the Teensy board is running the LED blink, or not connected, you should see this window.

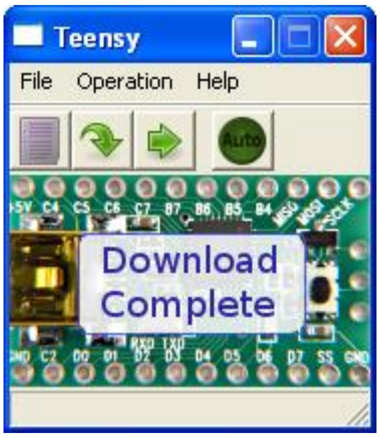


If HalfKay is running, you should see this. If not, simply make sure your Teensy board is connected, and press the pushbutton to run HalfKay. The Teensy Loader will quickly recognize it.

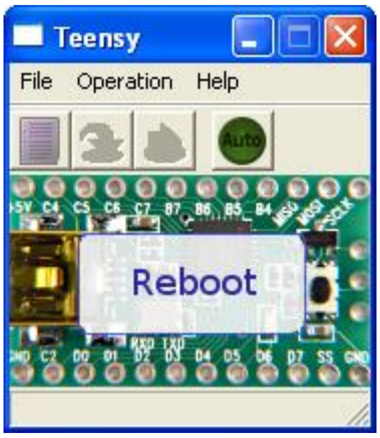


From the File name, choose "Open HEX File" and open blink_fast.hex, which you downloaded using the link at the top of this page. You can also open the file using the toolbar button, or using drag-and-drop onto the upper portion of the window. When the file is opened, the filename and percentage usage of the Teensy's memory are shown.

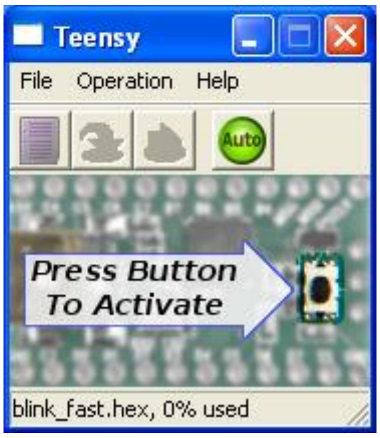
Select "Program" from the "Operations" menu, or click the Program button on the tool bar. The download should be very quick, perhaps too fast to see a tiny progress bar appear, but you should see the "Download Complete" message.



Choose "Reboot" from the "Operations" menu, or click the Reboot button on the tool bar. Your Teensy board should immediately begin running the fast LED blink program!



To try Automatic mode, simply click the "Auto" button, or choose "Automatic Mode" from the Operations menu. The Auto button should illuminate bright green. Then, open the slow LED blink HEX file.



When you press the button on your Teensy board, Automatic Mode will quickly program and reboot your board. Normally, while working on a project, you can leave the Teensy Loader in Automatic Mode. Teensy Loader always reads the latest version of your HEX file, so you can just compile your code, then press the pushbutton to program and run your code.

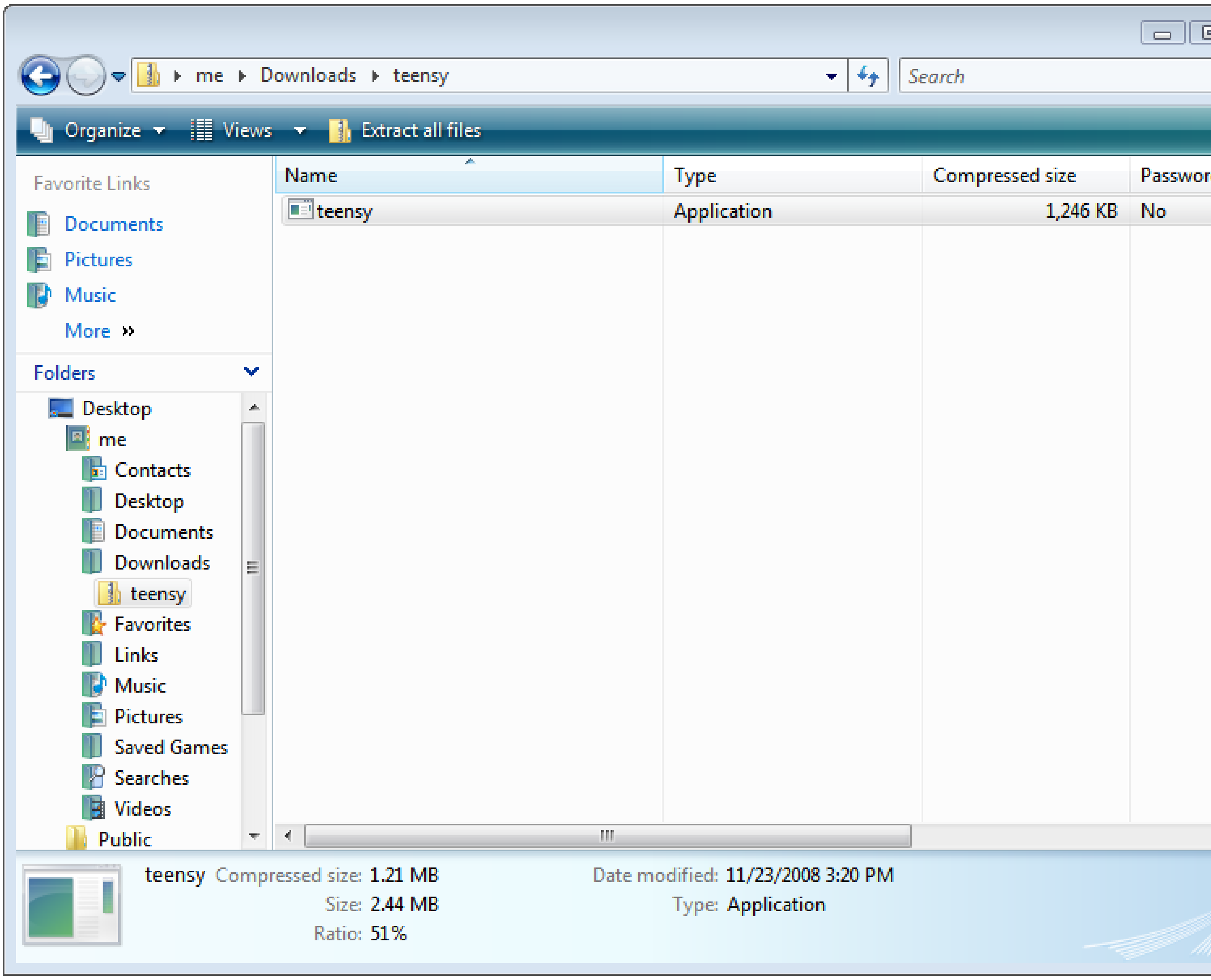
Using The Teensy Loader on Windows Vista

Update: Teensy 3.0 is supported by Teensyduino. After installation, Teensy Loader is in hardware/tools. It is run automatically when using Verify or Upload within the Arduino software. This version on this web page has not been updated for Teensy 3.0. Please insta ll Teensyduino to use Teensy 3.0.

Files to download:

- Teensy Loader Program (1.3 megabytes)
- LED Blink, Both Slow & Fast

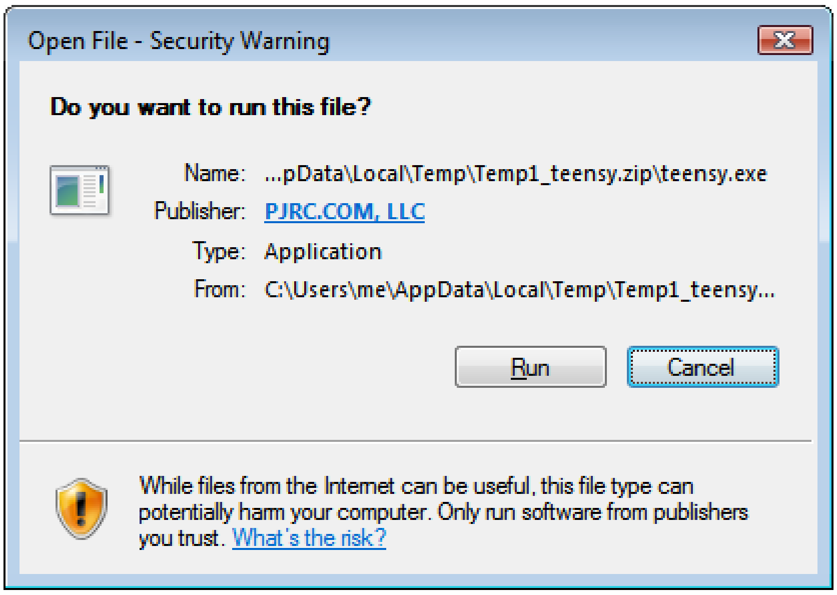
TEENSY.EXE is a single file application, so there is no installer required. You may copy it anywhere on your computer.



Name	Type	Compressed size	Passwor
teensy	Application	1,246 KB	No

teensy Compressed size: 1.21 MB
Size: 2.44 MB
Ratio: 51%
Date modified: 11/23/2008 3:20 PM
Type: Application

Simply double click the application to run it. The first time it runs, Microsoft will ask you to confirm. This window should show "PJRC.COM, LLC" as the publisher.

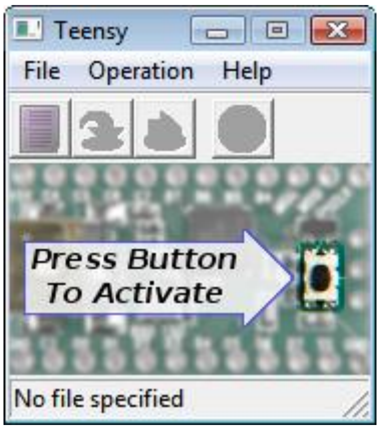

Open File - Security Warning

Do you want to run this file?

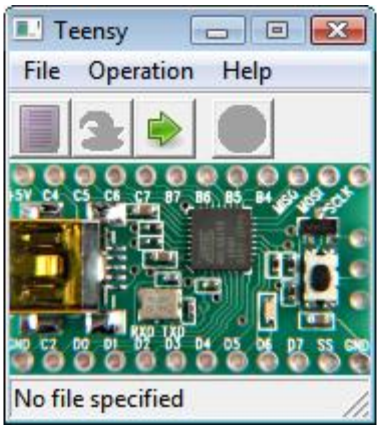
Name: ...pData\Local\Temp\Temp1_teensy.zip\teensy.exe
Publisher: PJRC.COM, LLC
Type: Application
From: C:\Users\me\AppData\Local\Temp\Temp1_teensy...

While files from the Internet can be useful, this file type can potentially harm your computer. Only run software from publishers you trust. What's the risk?

The Teensy Loader should appear as a small window. If the Teensy board is running the LED blink, or not connected, you should see this window.

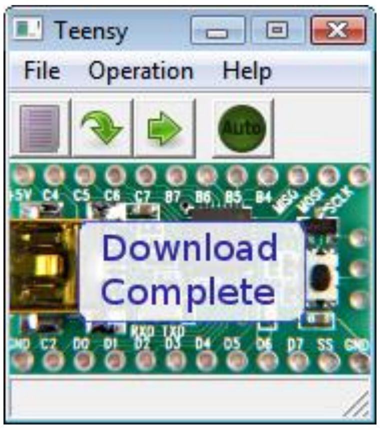


If HalfKay is running, you should see this. If not, simply make sure your Teensy board is connected, and press the pushbutton to run HalfKay. The Teensy Loader will quickly recognize it.

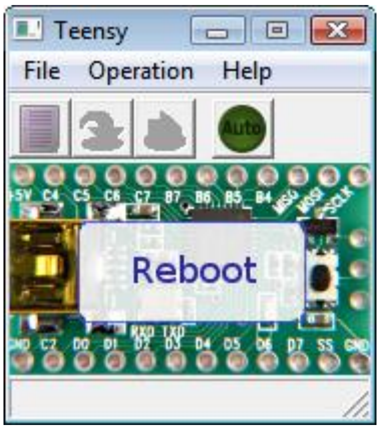


From the File name, choose "Open HEX File" and open blink_fast.hex, which you downloaded using the link at the top of this page. You can also open the file using the toolbar button, or using drag-and-drop onto the upper portion of the window. When the file is opened, the filename and percentage usage of the Teensy's memory are shown.

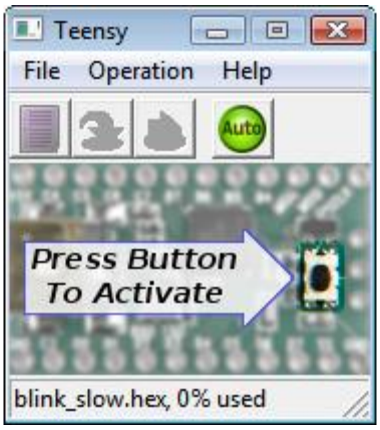
Select "Program" from the "Operations" menu, or click the Program button on the tool bar. The download should be very quick, perhaps too fast to see a tiny progress bar appear, but you should see the "Download Complete" message.



Choose "Reboot" from the "Operations" menu, or click the Reboot button on the tool bar. Your Teensy board should immediately begin running the fast LED blink program!



To try Automatic mode, simply click the "Auto" button, or choose "Automatic Mode" from the Operations menu. The Auto button should illuminate bright green. Then, open the slow LED blink HEX file.



blink_slow.hex, 0% used

When you press the button on your Teensy board, Automatic Mode will quickly program and reboot your board. Normally, while working on a project, you can leave the Teensy Loader in Automatic Mode. Teensy Loader always reads the latest version of your HEX file, so you can just compile your code, then press the pushbutton to program and run your code.

Teensy Loader, Command Line

The Teensy Loader is available in a command line version for advanced users who want to automate programming, typically using a Makefile. For most uses, the graphical version in Automatic Mode is much easier.

Files to download:

- Teensy Loader Command Line Source (github)
- Linux: udev rules for Teensy
- FreeBSD: device conf for Teensy

Compiling From Source

The command line version is provided as source code for most platforms. To compile, you must have gcc or mingw installed. Edit the Makefile to select your operating system, then just type "make". If you have a BSD compatible make, replace "Makefile" with "Makefile.bsd".

Version 2.0 has been tested on Ubuntu 9.04, Mac OS-X 10.5, Windows XP, FreeBSD 8.0, OpenBSD (20-Jan-2010 snapshot), and NetBSD 5.0.1. All versions of NT-based Windows with USB support (2000 and later) are believed to work.

On Ubuntu, you may need to install "libusb-dev" to compile.

```
  sudo apt-get install libusb-dev
```

Usage and Command Line Options

A typical usage from the command line may look like this:

```
teensy_loader_cli -mmcu=atmega32u4 -w blink_fast.hex
```

Required command line parameters:

-mmcu=<MCU> : Specify Processor. You must specify the target processor. This syntax is the same as used by gcc, which makes integrating with your Makefile easier. Valid options are:

-mmcu=mk20dx256 :	Teensy 3.1 (linux & mac only)
-mmcu=mk20dx128 :	Teensy 3.0 (linux & mac only)
-mmcu=mkl26z64 :	Teensy-LC (linux & mac only)
-mmcu=atmega32u4 :	Teensy 2.0
-mmcu=at90usb1286 :	Teensy++ 2.0
-mmcu=at90usb162 :	Teensy 1.0
-mmcu=at90usb646 :	Teensy++ 1.0

Caution: HEX files compiled with USB support must be compiled for the correct chip. If you load a file built for a different chip, often it will hang while trying to initialize the on-chip USB controller (each chip has a different PLL-based clock generator). On some PCs, this can "confuse" your USB port and a cold reboot may be required to restore USB functionality. When a Teensy has been programmed with such incorrect code, the reset button must be held down BEFORE the USB cable is connected, and then released only after the USB cable is fully connected.

Optional command line parameters:

-w : Wait for device to appear. When the pushbuttons has not been pressed and HalfKay may not be running yet, this option makes teensy_loader_cli wait. It is safe to use this when HalfKay is already running. The hex file is read before waiting to verify it exists, and again immediately after the device is detected.

-r : Use hard reboot if device not online. Perform a hard reset using a second Teensy running this rebootor code, with pin C7 connected to the reset pin on your main Teensy. While this requires using a second board, it allows a Makefile to fully automate reprogramming your Teensy. No manual button press is required!

-s : Use sort reboot if device not online. Perform a sort reset request by searching for any Teensy running USB Serial code built by Teensyduino. A request to reboot is transmitted to the first device found.

-n : No reboot after programming. After programming the hex file, do not reboot. HalfKay remains running. This option may be useful if you wish to program the code but do not intend for it to run until the Teensy is installed inside a system with its I/O pins connected.

-v : Verbose output. Normally teensy_loader_cli prints only error messages if any operation fails. This enables verbose output, which can help with troubleshooting, or simply show you more status information.

System Specific Setup

Linux requires UDEV rules for non-root users.

FreeBSD requires a device configuration file for non-root users.

OpenBSD's make is incompatible with most AVR makefiles. Use "pkg_add -r gmake", and then compile code with "gmake all" to obtain the .hex file.

On Macintosh OS-X 10.8, Casey Rodarmor shared this tip:

I recently had a little trouble getting the teensy cli loader working on Mac OSX 10.8. Apple moved the location of the SDKs around, so that they now reside inside of the xcode app itself. This is the line in the makefile that got it working for me:

```
SDK ?=
/Applications/Xcode.app/Contents/Developer/Platforms/MacOSX.platform/Developer/SDKs/MacO
SX10.8.sdk
```

Makefile Integration

You can use teensy_loader_cli from your Makefile, to autoamtically program your freshly compiled code. Here is an example:

```
# Create final output files (.hex, .eep) from ELF output file.
%.hex: %.elf
	@echo
	@echo $(MSG_FLASH) $@
	$(OBJCOPY) -O $(FORMAT) -R .eeprom -R .fuse -R .lock -R .signature $< $@
	teensy_loader_cli -mmcu=$(MCU) -w -v $@
```

Make requires the white space before any command to be a tab character (not 8 spaces), so please make sure you use tab.

If you connect a second Teensy using the rebootor code, add the "-r" option and your code will always be programmed automaticallly without having to manually press the reset button!

Scott Bronson contributed a Makefile patch to allow "make program" to work for the blinky example.

PlatformIO Integration

http://platformio.org

Errata

Compiling on Mac OS-X 10.6 may require adding "-mmacosx-version-min=10.5" to the Makefile. Thanks to Morgan Sutherland for reporting this.

Step #3: The GCC Compiler and Tools

Update: This page only applies to Teensy 2.0. Teensy 3.0 is supported by Teensyduino. To use Teensy 3.0 without Arduino, first install Teensyduino. A sample makefile is installed in hardware/teensy/cores/teensy3. The makefile comments explain how to detete the unnecessary portions of Arduino, if you wish to use only the makefile.

To compile C and C++ programs, you will need a collection of programs including avr-gcc, the avr-libc C library and avr-binutils. Installing all the required pieces separately can be difficult. Fortunately, there are complete, easy to install packages for all major platforms.

Macintosh OS X

Download AVR MacPack. The MacPack disk image has an installer that does everything for you. The following command may need to be entered (in Terminal) to choose the version of gcc which works with Teensy.

```
avr-gcc-select 4
```

Linux, Ubuntu 8.10

Ubuntu provides packages, so you can just install them using this command.

```
sudo apt-get install gcc-avr binutils-avr avr-libc
```

Windows XP & Vista & 7 & 8

Download WinAVR, which includes everything you need, with a nice installer.

Using Make to Compile Blinky

Download: Blinky Example Source Code (zip file)

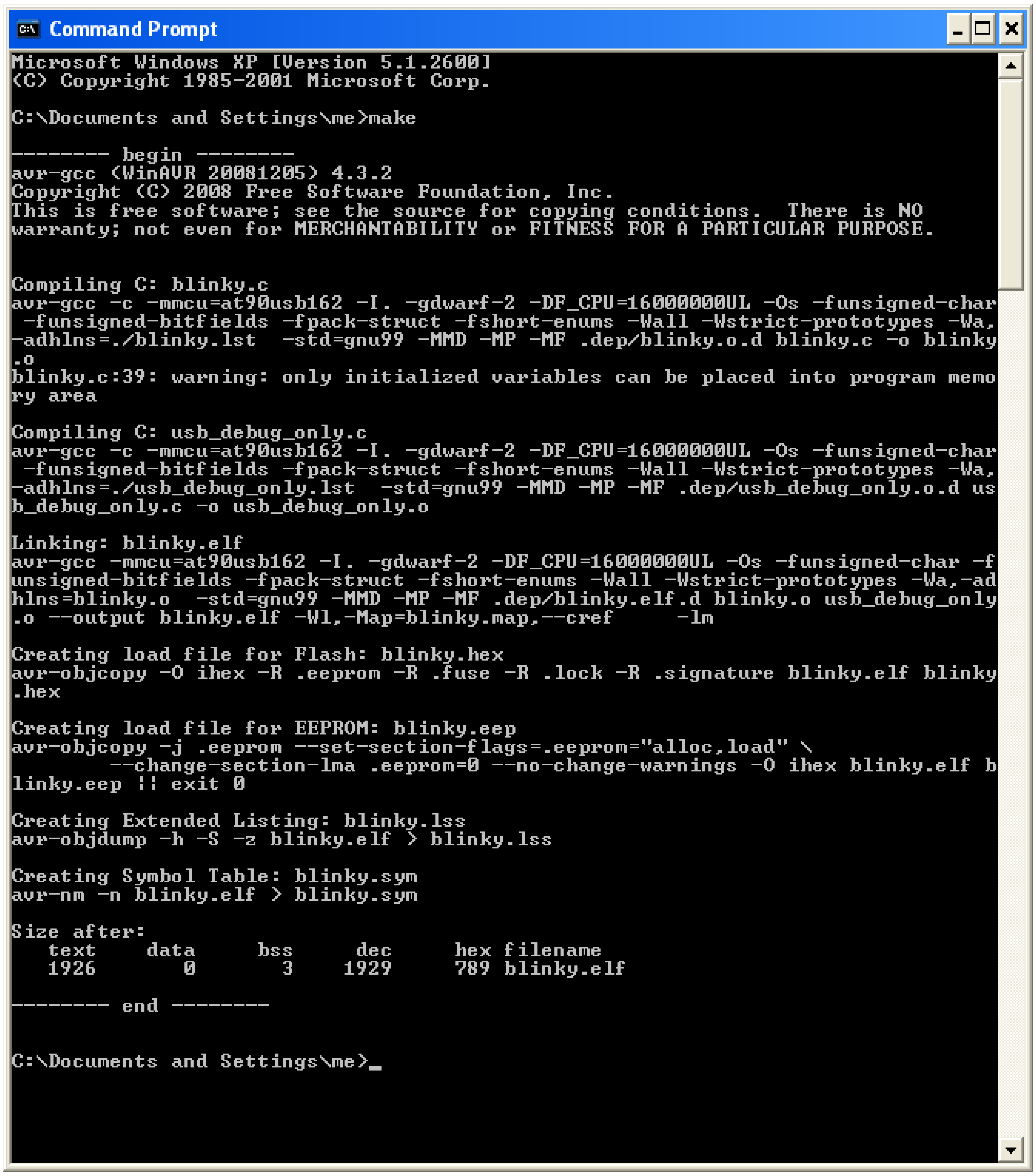
Now that you have the compiler installed, a first step is to compile the blinky source into a .HEX file you can put onto your Teensy using the Teensy Loader. Of course, if you skipped the previous step, you'll need to install the Teensy Loader to make use of the .HEX file created here.

Make is a command line program that automatically runs the compiler and other related programs to perform all the steps necessary to compile your code into the .HEX file. Fortunately, make is very simple to use. At the command prompt, just type "make".

On Windows XP or Vista, you need to run "Command Prompt" which is in the Start menu under All Programs -> Accessories. On Mac OS X, run Terminal, which is found in Applications/Utilities. Linux systems have many ways to access the command line. On all systems, the command line works in basically the same way.

The command line interface accesses a single directory, which is usually shown in the prompt. You can type the "cd" command to change this directory. If you find typing commands cumbersome, just take notice of which directory is used and place the blinky files in that location. For example, in the screenshot below the directory is "**C:\Documents and Settings\me**" and the files from blinky.zip were just copied to that location.

Just type **make** and it will do everything and give you detailed information.



Command Prompt

```
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\me>make

-------- begin --------
avr-gcc (WinAVR 20081205) 4.3.2
Copyright (C) 2008 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.


Compiling C: blinky.c
avr-gcc -c -mmcu=at90usb162 -I. -gdwarf-2 -DF_CPU=16000000UL -Os -funsigned-char
 -funsigned-bitfields -fpack-struct -fshort-enums -Wall -Wstrict-prototypes -Wa,
-adhlns=./blinky.lst  -std=gnu99 -MMD -MP -MF .dep/blinky.o.d blinky.c -o blinky
.o
blinky.c:39: warning: only initialized variables can be placed into program memo
ry area

Compiling C: usb_debug_only.c
avr-gcc -c -mmcu=at90usb162 -I. -gdwarf-2 -DF_CPU=16000000UL -Os -funsigned-char
 -funsigned-bitfields -fpack-struct -fshort-enums -Wall -Wstrict-prototypes -Wa,
-adhlns=./usb_debug_only.lst  -std=gnu99 -MMD -MP -MF .dep/usb_debug_only.o.d us
b_debug_only.c -o usb_debug_only.o

Linking: blinky.elf
avr-gcc -mmcu=at90usb162 -I. -gdwarf-2 -DF_CPU=16000000UL -Os -funsigned-char -f
unsigned-bitfields -fpack-struct -fshort-enums -Wall -Wstrict-prototypes -Wa,-ad
hlns=blinky.o  -std=gnu99 -MMD -MP -MF .dep/blinky.elf.d blinky.o usb_debug_only
.o --output blinky.elf -Wl,-Map=blinky.map,--cref     -lm

Creating load file for Flash: blinky.hex
avr-objcopy -O ihex -R .eeprom -R .fuse -R .lock -R .signature blinky.elf blinky
.hex

Creating load file for EEPROM: blinky.eep
avr-objcopy -j .eeprom --set-section-flags=.eeprom="alloc,load" \
        --change-section-lma .eeprom=0 --no-change-warnings -O ihex blinky.elf b
linky.eep || exit 0

Creating Extended Listing: blinky.lss
avr-objdump -h -S -z blinky.elf > blinky.lss

Creating Symbol Table: blinky.sym
avr-nm -n blinky.elf > blinky.sym

Size after:
   text    data     bss     dec     hex filename
   1926       0       3    1929     789 blinky.elf

-------- end --------


C:\Documents and Settings\me>_
```

That is a lot of information! Really, the most important is what is NOT shown, any errors that prevent make from completing the job. Normally when you run make, if you see the size summary at the end, you've got a .HEX file and you just reach over and press the pushbutton and (if Teensy Loader is in Auto mode) within a few seconds your freshly compiled code starts running!

Blinky blinks messages in Morse Code on the LED, so you should see patterns of Morse Code messages when the Blinky starts running.

The size summary is usually worth reading. In this example, "text" is 1926 bytes, "data" is 0, and "bss" is 3 bytes. The "dec" and "hex" are just the sum of these other numbers. So what to these names really mean?

Section	**Meaning**
text	Compiled C code, in Flash memory
data	Variables that get initialized at startup, Flash and RAM
bss	Variables that are zero at startup, RAM only

Just add the text and data numbers to get the total flash memory used. The Teensy Loader also shows the percentage of the chip your HEX file consumes.

RAM usage is more difficult to understand. The data and bss variables are placed at the beginning of the RAM. Local variables, function call return addresses and temporary memory used by interrrupt routines are allocated on a stack that starts at the highest RAM address and grows downward, towards your variables. There is no easy way to know exactly how much memory the stack will use.

Understanding the Makefile

When you run make, it reads a file named "Makefile" for instructions to build your code. The Makefile can be edited with any text editor. It is long and complicated, but fortunately only a few very simple settings need to be edited for most projects.

Near the beginning of the file, you will find these definitions. TARGET is the name of your project. You may want a better name than "blinky".

```
# Target file name (without extension).
TARGET = blinky
```

SRC is the list of all your source files. There are generally 3 reasons to structure your program into different files.

1. Related functions can be grouped into separate files, usually so you do not need to bother seeing them as you edit other code.
2. Make detects which files have changed and only recompiles those, reducing total compile time for incremental changes.
3. Code you download from this site and others comes as separate C files. Adding them to your Makefile is the easiest way to use them. Especially if the C code is updated, keeping it separate from others makes replacing the old version much easier than if you copy-and-paste code into one giant file.

Usually #3 is the compelling reason!

```
# List C source files here. (C dependencies are automatically generated.)
SRC =   $(TARGET).c \
        usb_debug_only.c
```

As you add more files to your project, just include each one in this list. The backward slash "\" at the end of a line means the SRC list continues on the next line, so be sure every line except the last ends in a backward slash.

Usually each .c file will have a .h header that you use with #include in the other .c files that will call its functions or access its variables. The "C dependencies are automatically generated" means you to NOT need to specify the .h files and which .c files include them. Make will examine your code and learn all those dependencies automatically.

Details In The Blinky Source Code

Near the top of blinky.c are these definitions, for using the LED.

```
#define LED_CONFIG       (DDRD |= (1<<6))
#define LED_ON           (PORTD &= ~(1<<6))
#define LED_OFF          (PORTD |= (1<<6))
```

PORTD and DDRD are the actual hardware registers that control the port D pins, and these definitions provide easy-to-read names for the LED. See the using I/O pins page for details.

Several definitions follow, including some unusual string syntax.

```
void morse_string_P(const char *s);
#define morse(s)        morse_string_P(PSTR(s))
```

Normally, all variables (even string constants) are placed in RAM. But there isn't much RAM available and string constants are rather wasteful, because a copy would be in flash anyway to initialize the RAM at startup! The PSTR macro causes the string to be placed in flash memory. By convention, functions ending in "_P" do special operations to read from flash instead of RAM.

Blinky's main loop is very simple. It will just blink "SOS" 6 times, then blink a long question, and repeat forever. The _delay_ms() function is provided by the C library.

```
        while (1) {
                for (i=0; i<6; i++) {
                        morse("SOS");
                        _delay_ms(1500);
                }
                morse("DOES ANYBODY STILL KNOW MORSE CODE?");
                _delay_ms(4000);
        }
```

Inside the morse_character() function, which blinks a single character in Morse Code, there are calls to usb_debug_putchar() and print(), which just calls usb_debug_putchar().

```
        if (c < 'A' || c > 'Z') {
                print("Opps, unsupported character: ");
                usb_debug_putchar(c);
                print("\n");
                return;
        }
```

The HID Listen program is used to receive and display all these characters. If your Teensy is blinking Morse Code, now is the time to install HID Listen to see these messages.

Step #4: The HID Listen Program

Update: This page only applies to Teensy 2.0. Teensy 3.0 is supported by Teensyduino. To use Teensy 3.0 without Arduino, first install Teensyduino. A sample makefi le is installed in hardware/teensy/cores/teensy3. The makefile comments explain how to detete the unn ecessary portions of Arduino, if you wish to use only the makefile.

In the previous step you compiled and loaded blinky.hex onto your Teensy board. While it's blinking Morse Code, blinky also sends characters by calling usb_debug_putchar(), which you can display with the HID Listen program. Other examples use these same functions, so HID Listen is a valuable tool for "seeing" what your code is doing.

If you didn't compile blinky, you can also find a copy of blinky.hex in the source code package here.

Download HID Listen

- Macintosh OS X executable
- Linux executable (32 bit only)
- Windows executable
- Source Code + all 3 executables + blinky.hex

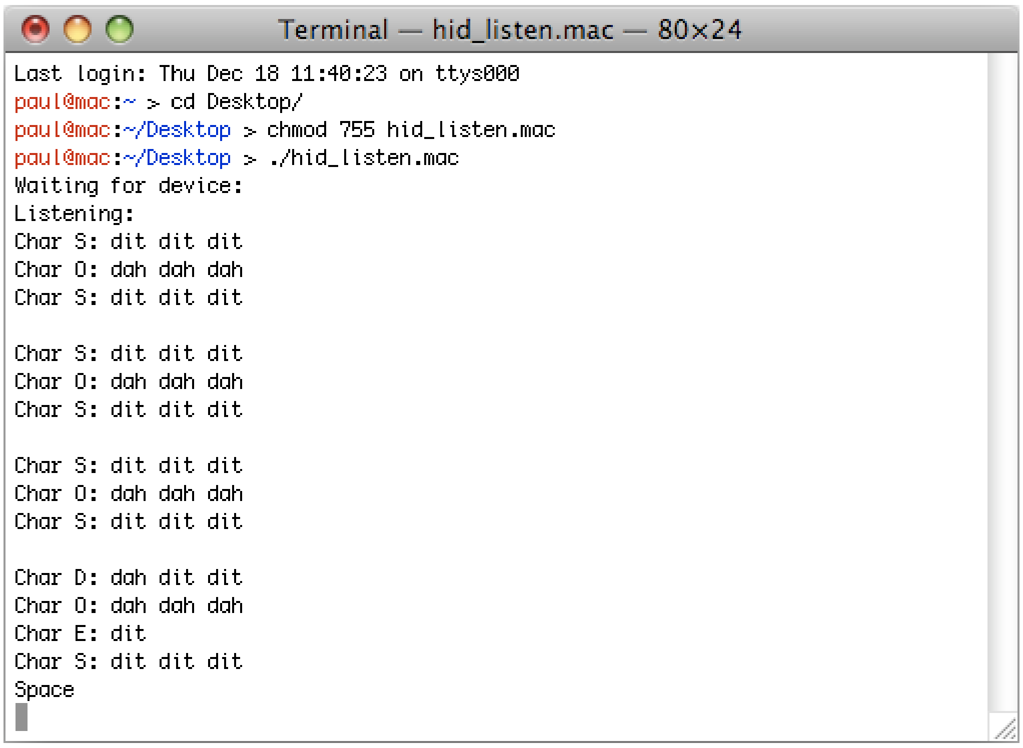
No driver installation is required. All major operating systems have the USB HID drivers built in.

Running HID Listen

HID Listen is a command line program, so you need to run it from the command line. A GUI-based version is planned, but for now the only option is this simple command line version.

Macintosh OS X 10.5

Just run the hid_listen.mac program in Terminal. You may need to give the file execute permission with the "chmod" command before you can run it.

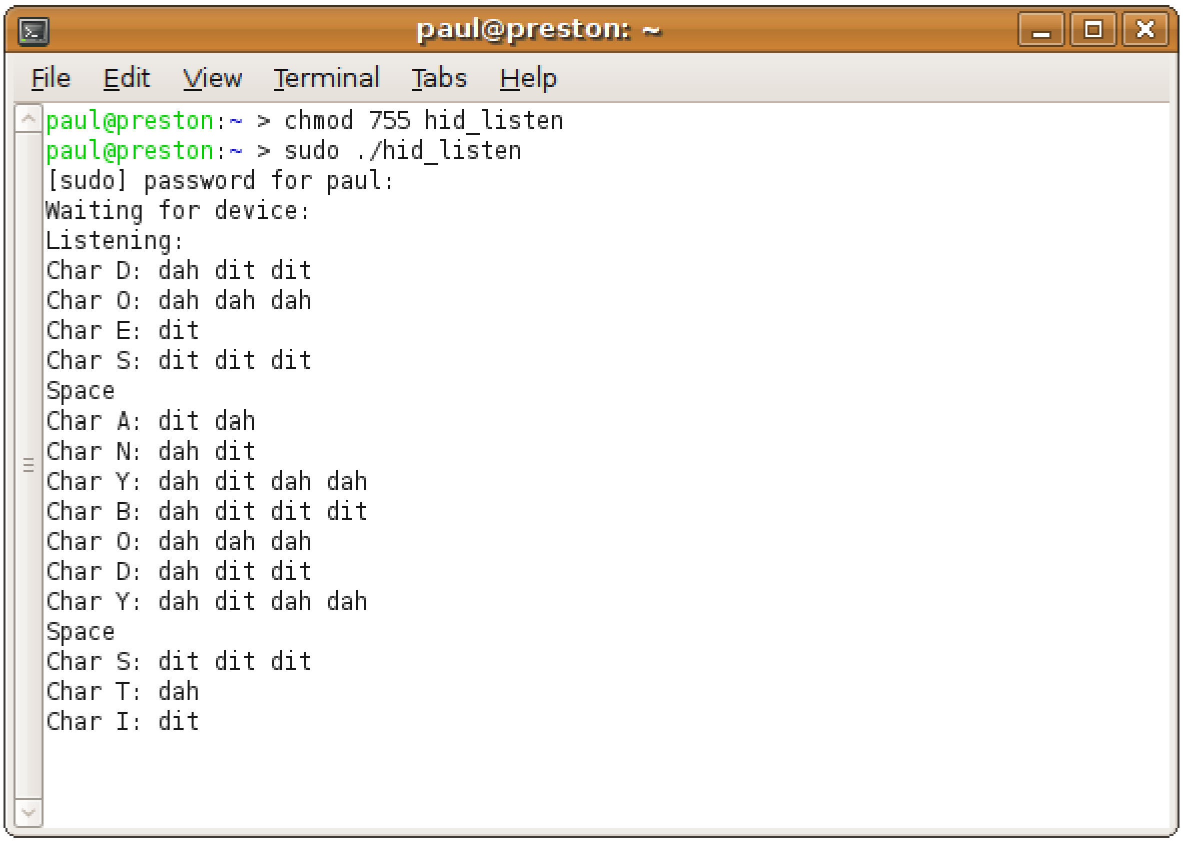


```
Last login: Thu Dec 18 11:40:23 on ttys000
paul@mac:~ > cd Desktop/
paul@mac:~/Desktop > chmod 755 hid_listen.mac
paul@mac:~/Desktop > ./hid_listen.mac
Waiting for device:
Listening:
Char S: dit dit dit
Char O: dah dah dah
Char S: dit dit dit

Char S: dit dit dit
Char O: dah dah dah
Char S: dit dit dit

Char S: dit dit dit
Char O: dah dah dah
Char S: dit dit dit

Char D: dah dit dit
Char O: dah dah dah
Char E: dit
Char S: dit dit dit
Space
```

Linux

On Ubuntu Linux, the /dev/hidraw device files are not readable by default for normal users. You could create a custom udev rule, or simply run hid_listen with sudo.

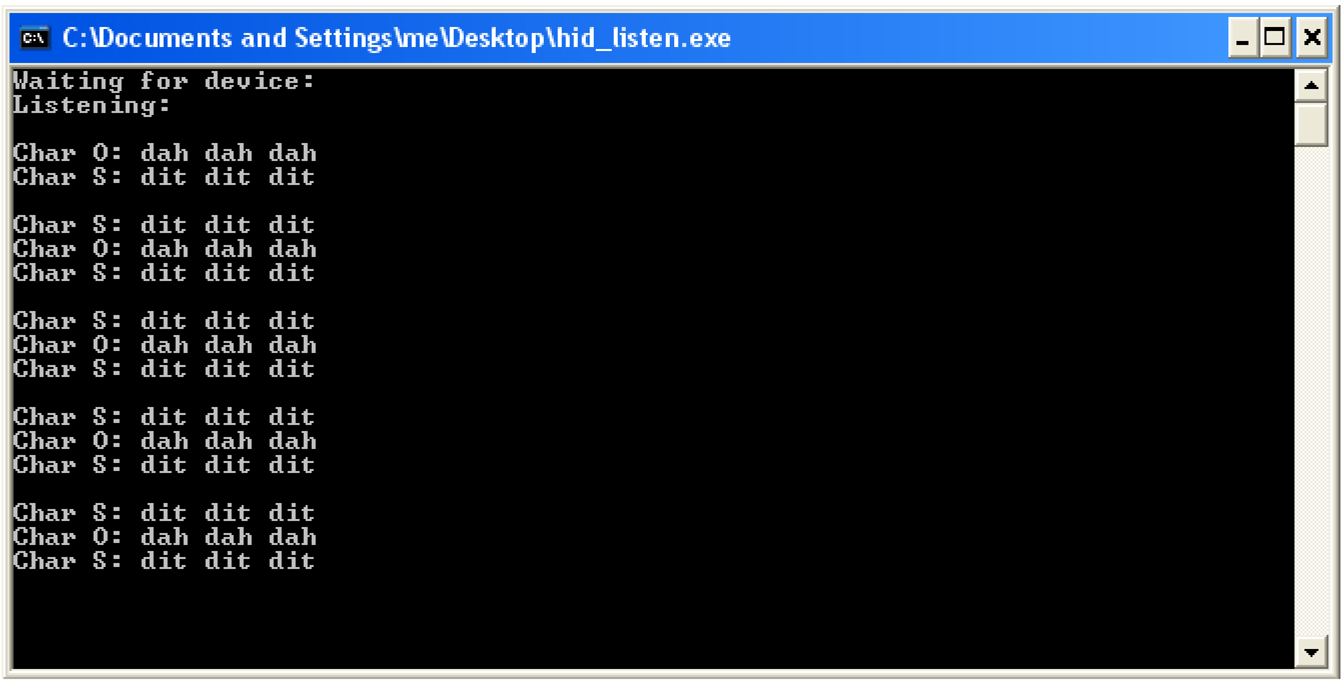
Update, Jan 1, 2011: The 32 bit linux binary does not work on 64 bit linux systems (2.6.32 tested), likely because of missing hidraw 32 bit ioctl support (HIDIOCGDEVINFO, HIDIOCGVERSION, HIDIOCAPPLICATION). Future kernels may fix this. Until then, hid_listen must be recompiled from the source to work on 64 bit linux.



```
paul@preston:~ > chmod 755 hid_listen
paul@preston:~ > sudo ./hid_listen
[sudo] password for paul:
Waiting for device:
Listening:
Char D: dah dit dit
Char O: dah dah dah
Char E: dit
Char S: dit dit dit
Space
Char A: dit dah
Char N: dah dit
Char Y: dah dit dah dah
Char B: dah dit dit dit
Char O: dah dah dah
Char D: dah dit dit
Char Y: dah dit dah dah
Space
Char S: dit dit dit
Char T: dah
Char I: dit
```

Windows XP & Vista

On Windows, simply double click the hid_listen.exe program. Windows will automatically launch the command prompt and run hid_listen in it.



```
Waiting for device:
Listening:

Char O: dah dah dah
Char S: dit dit dit

Char S: dit dit dit
Char O: dah dah dah
Char S: dit dit dit

Char S: dit dit dit
Char O: dah dah dah
Char S: dit dit dit

Char S: dit dit dit
Char O: dah dah dah
Char S: dit dit dit

Char S: dit dit dit
Char O: dah dah dah
Char S: dit dit dit
```