The KeeLoggers

hardware keyboard logger devices

Try a **hardware keylogger** instead of a software keyboard logger! This **hardware keyboard recorder** is **system independent**, **undetectable for software**, and **installs in seconds** (even with the PC off).

products
- KeeLogger
- KeeLogger Pro
- KL Modules
- KeeVirus

documents
- F.A.Q.
- Install guide
- Software
- PC keyboard
- Do-It-Yourself
- Specifications
- Misc. issues
- Links

ordering
- Prices
- Payment
- Ordering form
- Contact

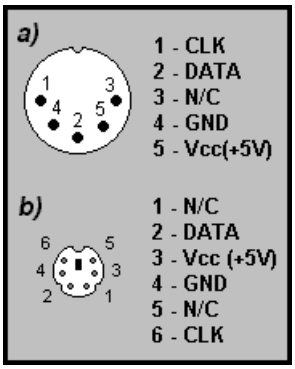
features
- Quick install

- New software

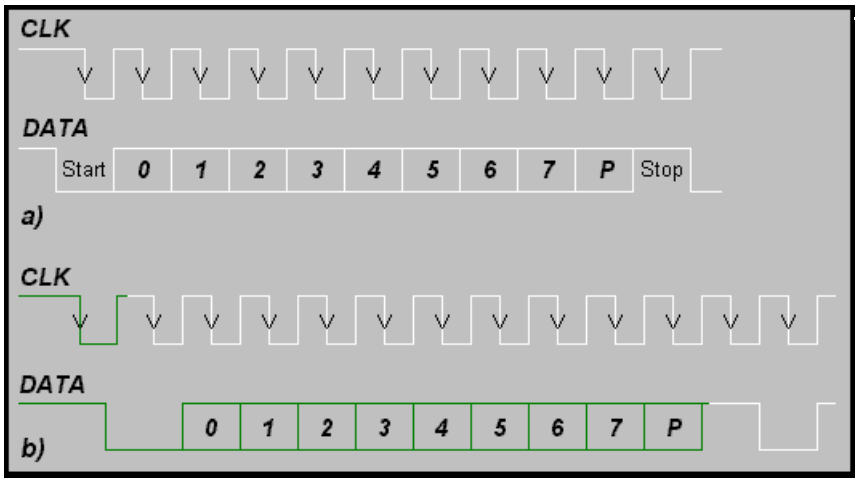
KLD v3.0 now with national keyboard layouts and new search options!

The PC keyboard protocol

The background



If you were to cut a PC keyboard cable through, you would probably find 6 wires within. Only 4 of these are meaningful. Two of these are power lines : ground (GND) and +5 volt from the computer power supply (Vcc). The other two wires are synchronous transmission lines: the data line (DATA) and the clock line (CLK). You can see how these lines correspond to DIN (a) and PS/2 (b) connector pins on the figure to the right. Transmission is bi-directional, however the keyboard is superior. The keyboard sends information about keys which have been pressed and released. The data chunk consists of only one byte, preceded with a starting bit, and followed by a parity and stop bit. The keyboard puts successive bits on the DATA line, and clocks them with negative impulses on the CLK line. Clock frequency is 10...30kHz. This would be a very nice serial protocol if it wasn't for the computer, which occasionally wants to send information to the keyboard. In such cases, the PC pulls the CLK line low for some time and waits for the keyboard to start generating impulses. When these impulses start, it clocks it's own character in on the DATA line. You can see state diagrams of keyboard to host (a) and host to keyboard (b) transmission on the figure below. This protocol has of course a few exceptions, like interrupting a transmission, character repeat etc. However, these are very rare cases.



Data

So what is actually transmitted through the keyboard lines? On startup both the keyboard and the computer send initialization data, informing that they are OK. When the computer is running normally, only the keyboard sends data. This is data about every event that took place. An event is considered a key being pressed or released. If a standard key is pressed, its so called "scancode" is sent. Every key has exactly one scancode, creating a map of scancodes. If a key is released, first the special byte 240 is sent (0xf0 in hex), then the keys

New! KLD Lite for simple and quick data playback

misc

- Main page
- Download
- Resale
- Search
- Notifier

scancode is sent. So a standard keystroke causes 3 characters to be sent down the line. If a key is held down for some time, it's scancode will be generated constantly with the set repetition delay. When it's finally released, the 240 character will be sent, followed by the scancode.
This would still be a nice protocol if it wasn't for some special keys present on the standard PC keyboard, like Home, End, the arrows and so on. I really don't know, why the designers of this protocol did it this way. Over 250 characters can be generated using one byte, and the keyboard has slightly over 100 keys. This leaves plenty of "free" characters to use for additional keys. So what is the sense of introducing special keys? Whoever created this system must have had a bad day. Nevertheless, when a special key is pressed, the byte 224 (0xe0 in hex) is generated, then goes the scancode. When a special key is released, the sequence 224, 240 is fired (0xe0, 0xf0), followed by the scancode. All special keys are drawn on the map of scancodes with their 0xe0 characters beside them. OK, so this might still be easy. But there are also two super-special keys, Print Screen and Pause, which cause a whole bunch of data to be transmitted. For a keyboard interfacer, it's best to pretend these keys do not exist. See the links section for more information.

Logging

So how do the keyboard loggers work? Actually it's quite simple. A small microcontroller monitors the DATA and CLK lines all the time, acquiring all data. For the KeeLogger Pro in enhanced mode, data is logged to non-volatile EEPROM memory as it goes down the line. Thanks to this, the user can later find out about every event on the keyboard. This helps reconstructing the sequence in which a combination of keys was pressed and released. The standard KeeLogger does some interpretation of data going to the computer. The KeeLogger logs only key-presses for normal characters, and a few special characters, like Shift, Delete etc. This is to save memory and download time, as a lot of data is redundant.
When the user decides recording is over and types in the special password, the keyboard logger switches to playback mode. The keyboard gets switched off and the logger starts simulating key strokes from internal memory. The KeeLog Downloader application has to be active, to process the flow of data from the logger. Normal keys are simulated as they were written in memory, and special keys are transmitted using a two-byte hex code.