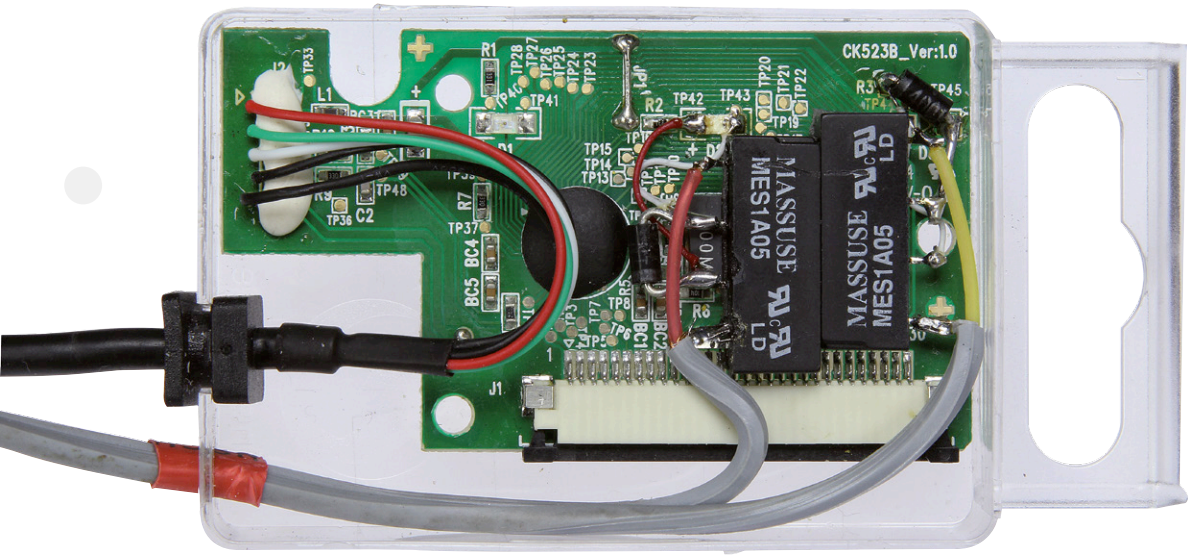
Output Signals from a Keyboard Controller

New use for an old keyboard

By **Joachim Berg**
(Germany)



There must be many PC users out there with an old USB keyboard kicking around in a junkbox somewhere. Maybe it's got a mechanical fault you haven't got round to fixing yet. No matter, this hack will turn it into something much more useful!

Take an old USB keyboard apart and you will find a small PCB containing a minimum of three status indicator LEDs for Number Lock, Caps Lock and Scroll Lock. With a little hardware modification and the help of software library functions we can use these to switch signals to the outside world. The computer operating system allows more than one keyboard to be attached at any one time so there is no reason why we can't plug a second keyboard into a spare USB port. In normal operation the Scroll Lock indicator is practically never used.

The only part you need from the keyboard is the controller PCB fitted on the right, under the number keypad. This contains the three status LEDs passing through the keyboard casing. As you can see in the title picture, it also has the USB cable attached. This consists of four wires providing 5 V power (red/black), data (green/white) and the screen.
If you are handy with a soldering iron this mod shouldn't take too long. The mini 5 V reed-relays used here draw so little operating current that they can just replace the LEDs on the board without the need for any extra buffers. The LED series resistors can also be left in-circuit to help reduce the load on the chip output transistors. The surface mount LEDs are no longer required so they can be unsoldered or ground off, make sure you don't damage the PCB pads; we need them to solder the wires to.
It's also possible to replace the LEDs with opto-couplers or even use larger 5 V relays

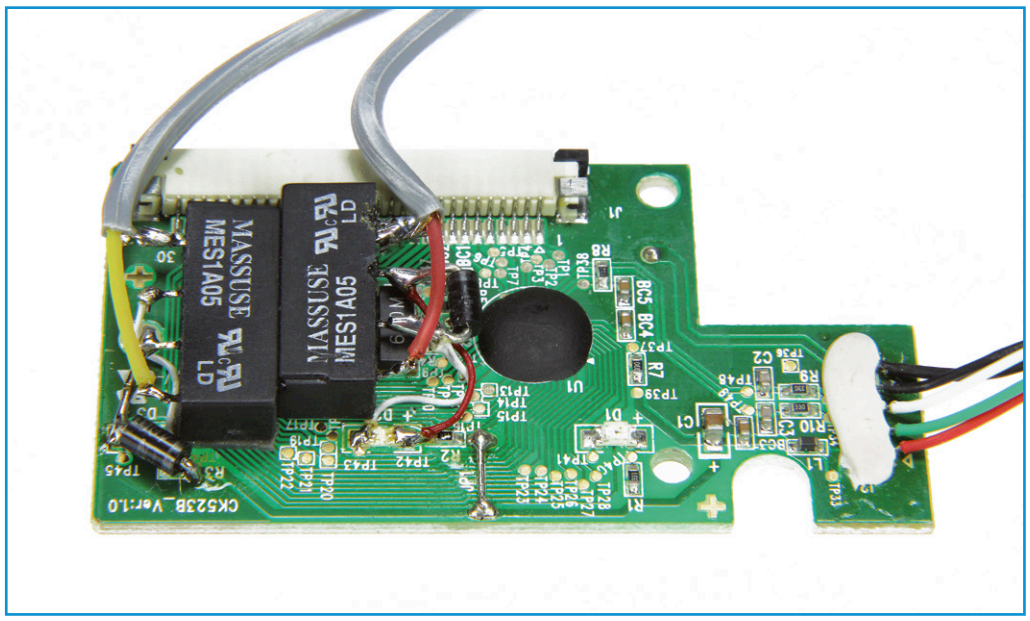


Figure 1.
The controller PCB with relays soldered in place.

that can switch line voltage but these will require a PNP transistor (or FET) with its base (or gate) connected directly to the LED connection. In this case you can just leave the LEDs in place. When choosing a different relay check that its coil impedance won't result in too much current being taken from the USB port. In general a USB device should not draw more than 500 mA, keyboards generally only have low power requirements.

The two additional components shown on the schematic are freewheel diodes (1N4001 or similar) connected across the relay coils to safeguard the driver stage in the controller chip (or any additional transistor driver) by snubbing the reverse-voltage induced when the relay coils are switched. A stereo cable has been soldered to the relay contacts but any other type of wire could also be used here. The completed circuit doesn't need a fancy enclosure; I've just used an old plastic case from a hardware store which once contained screws.

Software

Now we get to the interesting bit: I have written a small program library (.dll) which allows you to control the LEDs from any programming language. The source code is written in C# for .Net 4.0 or later and can be compiled with Microsoft Visual Studio™ [1]. The .dll is suitable for use by both 32-bit and 64-bit systems. The commentated source code (LdbLEDs.cs) and test applications can be found at [2]. **Table 1** lists the contents of the software package.

The **KbdLEDs.dll** library functions have been kept as simple as possible. Individual timers for the three switches are implemented without using multithreading.

The library has been compiled for *any CPU* so it is compatible at run time with both 64-bit and 32-bit platforms. It is only necessary to reference the .dll in your own program and then you will be able to use '*KbdLEDs.LED. ScrollLock'*, '*-.CapsLock'* and '*-.NumLock'* .

To change the state of a switch use '*Toggle(LED led)'*, using '*SetToggleState(LED led,*

Table 1. Software contents.

`KbdLEDs.dll`	The library
The `'Sources'`	Source code
`CallKbdLedsC.bat`	An example batch file for KbdLedsC
`KbdLedsC.exe`	Console program to use batch files
`TestKbdLEDs.exe`	Test program (WPF)
`TestKeyLEDsForms.exe`	Test program (Windows Forms)

bool toggled)' will cause the switch to be set and reset respectively. The switch status can be read using '*GetToggleState(LED led)'*. To produce a pulse use '*Impulse(LED led, int msDuration)*' with the pulse duration given in milliseconds. The LED will then be switched on for the defined period. The LED on times can overlap.

The **KbdLedsC.exe** console application provides a simple method to control the switches directly with a batch file. This gives an easy method to test their operation before integrating them into another program. The batch call can contain the following parameters:

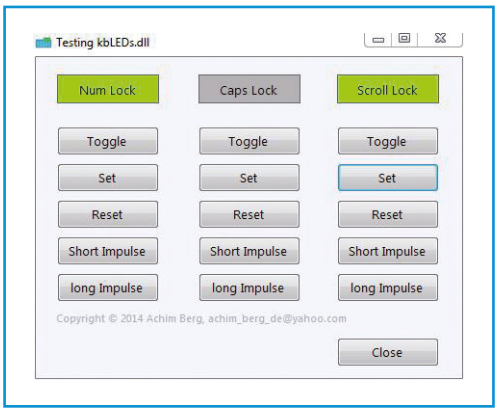


Figure 2.
The test program gives a quick impression of the .dll functions.

On/off switching

KbdLedsC /s=0 /s=1 /n=0 /n=1 /c=0 /c=1
(where s = Scroll Lock, n = Num Lock, c = Caps lock).

Pulse length

KbdLedsC /s=1000 /n=500 /c=2000
(time in milliseconds)

Several parameters can be passed in one call; the console application will issue a message if it is unhappy with the input.

The .dll library must be accessible; it can just be copied next to the console application. There are two common Frameworks in the .NET world which can be used to create graphical user interfaces (WPF and Windows Forms) and the download includes two such demo/test programs (**Figure 2**). Here the LED status is continually read (polled) which is OK just for the purposes of the test program but otherwise is poor programming style.

Some modern keyboards don't use an LED indicator for the scroll lock function but the demo program will still show the status of all three signals on the screen.

(140017)

Web Links

[1] www.visualstudio.com/

[2] www.elektor-magazine.com/post