Remote controlled: A computer without a keyboard

RAPID SWITCHING

MIRKO DÖLLE

Whether it's a fileserver or printer spooler: processors which have only one special function really need neither a monitor nor a keyboard – except that now and then a CD has to be changed or the system shut down at the weekend.

Everyone who has a server parked somewhere in the building certainly knows this problem: To change a CD you have to log on from a workstation, unmount, then go with the CD to the server and after changing it, back again to the workstation to mount. Or the 486 still in use as a printer spooler has to be powered down for the weekend – the fact that it has not yet been powered down is of course something you don't notice until after shutting down the last workstation. After a short time, monitor and keyboard get back together again so you don't have to keep running back and forth between workstation and server.

Viewed in the cold light of day, there are only a few actions such as shutdown,(un)mounting the CD drives or starting/continuing a backup, which you constantly come up against on the server. On the other hand the serial interface has various status lines which can be queried without going to a lot of trouble. A combination of the two produces a simple and very cheap remote control.

Inputs and outputs

A glance at Table 1 shows that in total, we have five inputs and three outputs. At first the send and receive lines (RXD, TXD) will remain unconnected. To connect one of the inputs we need a voltage. We could get this externally via a power supply or internally via a hard disk power point, but it is critical. The serial interface reacts sensitively to too high a voltage and in particular to too high a current. Since components nowadays are housed on the motherboard, is it not all that easy to exchange a damaged serial interface. Which is why we are going to use the outputs as power supply for the inputs. So there are four inputs left, one freely connectable output, for example for an LED, and a second, conditionally connectable output.

Wiring the inputs and outputs

The first output RTS will be used for a yellow LED, and this is connected with a series resistor between output and earth. We can use this for status messages or as acknowledgement for an action (on/off/flashing). The series resistor should be chosen such that not more than 20mA current flows, usually 2.2 kOhm.

The second output must be activated if it is to connect the inputs – So it is perfect as a system-status display. To do this we exploit the fact that the low-level of the serial interface is not earth (GND), but a negative voltage. If a series resistor and a two-pole dual-LED are placed between the output DTR and earth (polarity reversal of the LED causes it to change colour between red and green), then on red the buttons are inactive (because of negative voltage). The outputs are both low when the device is switched on, so at first the LED is red. After starting the control program (for example via an init-script) DTR is then set to high, the LED turns green and the inputs can be connected.

The four inputs CD, DSR, CTS and RI are connected, via four keys and a common series resistor of 10 kOhm with the output DTR. To avoid unintentional shutdown by touching a key, two red keys have been connected in series and linked to RI in the sample construction in Figure 1 – so both have to be pressed. The keys serve at the same time as a holder for the two LED's and for the sake of security are as far from each other as possible.

Control program

As described above, by activating the output RTS the yellow light diode is switched on. The control program published on our FTP server uses the LED as acknowledgement that a keypress has been detected and the associated action has been triggered. It is turned off again at the end of the action. If an error arises during this, the

yellow LED flashes. The dual LED at DTR acts as the readiness display. At the start of the control program it is switched to green, and at the end of the program to red.

The inputs are queried in a loop every 75ms. A key is regarded as pressed when the inputs remain unchanged during three runs (i.e. 225ms). The 75ms interval is set at random as a compromise between CPU load and reaction speed. The program can distinguish between individual keys and key combinations, and at present two actions are being implemented: If both red keys are pressed (input RI high), the yellow LED is set and the system is powered down (*halt*). If one or more white keys are pressed, then there is an unmount of */dev/cdrom* and the ejection or insertion and mounting on */cdrom*. For anyone for whom the algorithm for ejection and insertion is too imprecise, two keys can also be used. In total, seven actions, including combinations, can be triggered with the white keys. The red keys should be reserved for shutdown.

The C listing of the control program is provided with (hopefully) adequate comments, and even without more precise explanation of the source it should not be hard to adapt to local circumstances. Connected to a serial extension cable and with appropriately modified key configurations, one can even adjust the loudness of the sound card and control mpg123, if you ever leave the desk to listen to music.

Producing

It is not absolutely necessary to use the circuit board layout from Figure 2 - in principle it is also possible to simply screw and multiwire key switches and separate LED holders into a drive cover plate. The layout is intended for the key switches listed in the component list with and without LED holders, which can be obtained from places such as Maplin Electronics (*http://www.maplin.co.uk*). The only thing, which is important, is to be sure of the correct polarity of the two light diodes – and especially to watch for short circuits! The serial interfaces are usually located together with PCI bus and PS/2 ports in the southbridge directly on the motherboard. It would be tragic if this were to give up the ghost. Better to use a separate multi-IO card. This also means having no problem with the cable run, because you can come in via the PCB headers. It is also important to use serial resistors – the interface cannot usually handle more than 20mA per output. Connection is made either using a ribbon cable to a slot plate or through the casing aperture (ATX motherboards) from the outside, or via a PCB header directly on the plug of the motherboard or the interface card. Beware, the configuration of the PCB header is always different and should be looked up in the motherboard manual.

Figure 1: Sample construction with five keys

Future

By using a simple multiplexer it is possible to install seven white keys, instead of three, but this will mean no multiple combinations. It is also possible to add a display via an interface for additional information (which is why we have kept the RXD and TXD lines free), but this takes much longer and is more expensive and is for this reason being saved for another article. With the serial remote control described here it is possible to solve many everyday problems, for which one would otherwise have used a monitor and keyboard. ■

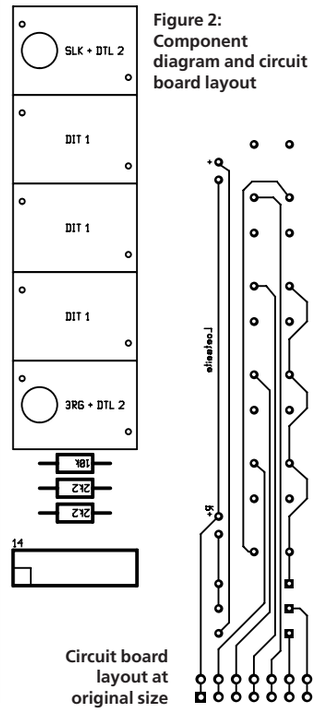

Figure 2: Component diagram and circuit board layout

Circuit board layout at original size

Table 1: Pin configuration of the serial interface

9 pin	25 pin	Direction	Signal	Designation	
1	8	input		CD	Carrier Detect
2	3	input		RXD	Receive Data
3	2	output	TXD	Transmit Data	
4	20	output	DTR	Data Terminal Ready	
5	7	-		GND	Ground
6	6	input		DSR	Data Set Ready
7	4	output	RTS	Request To Send	
8	5	input		CTS	Clear To Send
9	22	input		RI	Ring Indicator

Connection configuration of the sample circuit board (soldering lugs)

Pin	Signal	Component
1+2	GND	-
3+4	CD	white key switch, left
5+6	DSR	white key switch, middle
7+8	CTS	white key switch, right
9+10	RI	red key switch (series connection)
11+12	RTS	yellow LED
13+14	DTR	red/green LED

Component list

Quantity	Description
3	Key switch white
2	Key switch red with LED holder 3mm
1	LED 3mm yellow
1	Mini Bi-Colour LED 3mm red/green
2	resistor 2.2 kOhm
1	resistor 10 kOhm