The dBox IR-Keyboard

Table of contents

1. Revision history

Date	Description
2006-02-20	Initial version.
2006-03-29	Fixed some minor errors. Mention kb2rcd. Update and describe patch.

2. Introduction

Originally, the dBox was designed as a "multimedia terminal", with applications such as pay-per-view ordering, email, and home banking in mind. This design consideration amounts for the second (never officially used) card slot, the modem, and the optionally available infrared keyboard. This is all history, in particular, the keyboard was quite hard to get. Recently, they are available on German eBay for a quite reasonable price. (Search for "dbox tastatur").

The keyboard events are intercepted by the front processor of the dBox, making it possible both to read the keyboard as a generic computer keyboard, and to interpret the keys as coming from a normal remote control.

Some plugins access the keyboard. This topic is not covered by the present article, but may be found on Tuxbox WiKipedia, on the IR Keyboard page or on the Tuxbox commander page.

The empirical statements in this article has been verified on a Nokia, as well as on a Sagem dBox. It is believed that the Philips acts similarly.

The picture has been shamelessly stolen from Tuxbox WiKipedia.

3. The hardware

The keyboard is in laptop design and size, and comes exclusively in German qwertz-layout with German labeling. See the picture. There is a "fire button" on the extreme left, and a "joystick like" thing to the right. Below the "joystick", there is another "fire button". On the front, there are four IR-Diodes, that appear to provide quite reliable communication. I tried with a distance of 7 meters with no problems, probably much larger distances are possible.

The unit is powered by 4 AA-type batteries. It is suitable both for desktop- and sofa-usage. The mechanical quality is quite acceptable, in particular considering the low price.

4. Low-level interface

All keys send different infrared signals. The driver in `.../driver/fp/dbox2_fp_keyboard.c` translates these in keycodes and events. The keycodes are listed in the source code file `.../apps/tuxbox/neutrino/src/driver/rcinput.h`. These keycodes are in general from the Linux include file `include/linux/input.h`. The "Fn"-key to the lower left is made into a shift-key: it sends no keycode, but makes some keys (the ones having blue lettering) sending other keycodes. These are: KP0 to KP9, KPASTERISK, KPMINUS, KPPLUS, KPDOT, KPENTER (on the return key), KPSLASH. Until recently, the two Windows keys, and the key marked "Druck S-Abf" were left out.

As opposed to the remote control, no keys on the keyboard makes the dBox wake up from deep standby.

The left "fire button" sends the BTN_LEFT keycode, and the right one the BTN_RIGHT keycode. Usage of the "joystick" can be identified by the event type.

5. The keyboard as a computer keyboard

A front processor driver (`dbox2_fp_keyboard.o`) makes it possible to use the keyboard as a normal computer keyboard. With the command `loadkeys` (normally executed from `rcS`) it is possible to load a proper keyboard translation table. Since the keyboard is labeled as a German qwertz-keyboard, the `de-latin1-nodeadkeys` keymap is recommended. (`de-latin1`, used in the HEAD branch of CVS, is an alternative, however, for a computerist in general unsuitable.)

The appropriate keymap is loaded with the command like `loadkeys /share/keymaps/i386/qwertz/de-latin1-nodeadkeys.kmap.gz`. The

following files must be installed for this to work:

```
/bin/loadkeys
/share/keymaps/i386/qwertz/de-latin1-nodeadkeys.kmap.gz
/share/keymaps/i386/qwertz/de-latin1.kmap.gz
/share/keymaps/i386/include/linux-keys-bare.inc.gz
/share/keymaps/i386/include/linux-with-alt-and-altgr.inc.gz
/share/keymaps/i386/include/qwertz-layout.inc.gz
```

Around 100 kB space in the root partition is required. With this keyboard tables, everything (but the Windows keys) works, as far as I know, flawlessly.

Through `/etc/inittab`, there are more possibilities. With keys `Alt-F2` thought `Alt-F6` a virtual console is opened on the framebuffer. It is hidden (not closed) by the key `Alt-F1`. Finally, `/etc/inittab` instructs the system to reboot on the Cntrl-Alt-Del ("Strg-Alt-Enf" on German keyboards) key press.

6. The keyboard as a remote control for Neutrino

The first impression when trying the keyboard out of the box is that Neutrino recognizes very few of the keys. These are: The numerical keys 0 to 9, "Pos1" = Home, Bild ^ = Page Up, "Bild v" = Page Down (seldomly used, found only on very old remote controls), as well as the cursor keys. There is nothing wrong with this — the keyboard is a "keyboard", not a bulky replacement remote control!

In the menu `dBox -> Settings -> Key Setup`, it is possible to bind certain functionality (e.g. switching between TV- and radio-mode) to arbitrary keys. All keyboard keys (except for "Fn") can be used for this. With CVS from 2006-02-17 (2006-03-26 for the remaining three: KEY_SYSRQ, KEY_LEFTMETA, and KEX_RIGHTMETA), Neutrino also knows sensible names for those keys.

To be able to programmatically use the key, `rcinput.h` was extended. The keycodes have names taken from `include/linux/input.h`. From the keycodes, Neutrino key events are being named in an obvious manner.

Unfortunately, the event belonging to key on the German labeling denoted with ß, should logically be named `RC_minus`. However, this name was previously taken in a previous version of the file to denote the key for lowering the volume (`RC_volumedown` would have been better). For this reason, the name `RC_hyphen` was chosen.

6.1. Really turning the keyboard into a bulky remote control

Neutrino does not offer a clean way to, for example, add another "red button". So, instead the original version of this article presented a dirty way :-). (Just hard coding some translations into `rcinput.cpp`.)

6.2. The kb2rcd-daemon

Possibly as an answer, robspr1 in the Tuxbox forum wrote the daemon `kb2rcd`, see this posting (and following ones), as well as this thread in the Jack-the-Grabber forum. This is no doubt an interesting approach. It is a daemon that gets events (from `/dev/input/event0`), translates them, (not necessarily 1-1, but possibly 1-0 (Scripts), or 1-n (macros)) and pushes them back in the device. The advantage is the modularity, as it works with "everything", including plugins as well as Enigma. Also, it is easily added to an existing image, even cramfs/squashfs-Images. The drawback is that it works on the event-level; therefore everything like timing, up/down-Events etc. must be considered. As I tried it, at first it did not work at all, only for very long key presses. I found out that the initial delay (200ms) needed to be lowered. It can be configured using a configuration file `kb2rcd.conf`. It can execute commands, as well as plugins, directly. Recent versions can translate joystick actions to cursor keys, and can also interpret the Alt-key as a shift key, as well as assigning a binding to delayed keys. It is (partially) described in this Wiki-contribution (as well as the threads quoted). It has been checked in to CVS, in the directory `.../apps/tuxbox/tools/kb2rcd`.

6.3. A patch for rcinput

The patch for rcinput, available here, has been vastly improved. The translation is now governed by a configuration file, in spirit similar to the configuration file for `kb2rcd`. The patch works by translating keys, (not events), therefor no 1->n translation (macros) are possible. In particular, instead of key presses, several Neutrino-messages (see `.../apps/tuxbox/neutrino/src/neutrinoMessages.h`) can be generated, optionally with data. In this way, it is also possible to execute plugins directly .The configuration file should be located in `/var/tuxbox/conf/rc.conf`. A sample configuration file is shown in the Appendix.

To the file format of the configuration file: Every line is of the form *keyword=action* or *keyword=action(data)*. All other lines are ignored. A hash sign ("#") is taken as a comment character. Here *keyword* in general is the name of a key (translated to lower case), but there are also additional keywords:

Keyword	Allowed Values	Description
keyname (lower case)	*action* (lower case)	Have *action* be executed at press of key *keyname*.
`debug`	`on` and `off`	Turns on and off tracing on the system console.
`no_neutrinoevents_when_v`	`on` and `off`	If on, whenever a virtual console is visible (opened with F2 – F6), key presses will not be forwarded to neutrino.

Allowed actions are the key names, as well as some additional actions. These often, but not always, are the same as the corresponding Neutrino messages, translated to lower

case.

Action	Data	Description
keyname		Have Neutrino execute the action associated with *keyname*.
`system`	*command*	The data is taken as argument to a `system` command; in normal english, "is executed". Output is output to the system console. The return status is reported to the console.
`mode_tv`		Switches neutrino to TV mode
`mode_radio`		Switches neutrino to radio mode
`vcr_on`		Switches on SCART-mode.
`vcr_off`		Switches off SCART-mode.
`standby_on`		Switches on standby-mode.
`standby_off`		Switches off standby-mode.
`show_epg`		Shows the EPG.
`show_infobar`		Shows the infobar.
`lock_rc`		Locks the remote control (and the keyboard). Press the red key followed by the dBox key to re-enable. See this Wiki-article.
`show_volume`		Shows the volume bar.
`evt_popup`	*message*	Shows *message* in a temporary popup.
`evt_extmsg`	*message*	Shows *message* in an extmsg popup.
`evt_plugin`	*pluginname*`.cfg`	Starts the popup with the name *pluginname*.
`reload_conf`		Reload the configuration file.
`shutdown`		Shutdown the system. Note that this is a way of implementing a bona-fide discrete power-off-key.

7. Appendix. A sample rc.conf

```
# Demo rc.conf

A comment do not need a `#', as long as it does not contain an equal
sign

# Do not turn on debugging yet (it babbles while parsing this file),
# just at the end of the file
#debug=on

# Do not let Neutrino see the keys when a virtual console is open
no_neutrinoevents_when_vc=on

key_kp1=system(date)
key_kp2=system(ls)

# Projector suitable params (requires controldc)
key_kp5=system(controldc setVideoOutput 4) # YUV

# TV suitable params
key_kp8=system(controldc setVideoOutput 1;controldc setVideoFormat 0) #
RGB + auto
key_kp9=system(controldc setVideoFormat 1) # 16:9

key_minus=key_help
key_esc=key_home

key_f1=key_red
key_f2=key_green
key_f3=key_yellow
key_f4=key_blue
key_f5=mode_tv
key_f6=mode_radio
key_f7=vcr_on
key_f8=vcr_off
key_f9=standby_on
key_f10=standby_off
key_numlock=show_epg
key_sysrq=show_infobar
key_scrolllock=lock_rc
key_insert=show_volume

key_btnleft =key_ok
key_btnright=key_power
key_102nd=key_volumedown
key_grave=key_volumeup
key_pause=key_mute
key_delete=key_setup

evt_popup and evt_extmsg  works!
key_leftmeta=evt_popup(Barf rulez)
key_rightmeta=evt_extmsg(This nonsense stays on the screen for a LOOONG
time)

plugins can be started by evt_start_plugin
key_end=evt_start_plugin(tuxtxt.cfg)
```

```
special function: reload_conf reloads this file
key_tab=reload_conf

key_bottomright=shutdown

# NOW, turn on debugging
debug=on
```

8. Appendix. Installing the keyboard mapping file with newmake.

Here is a `root-local.sh` file that will install the above mentioned files in a newmake run. It violates this "style recommendation" severly ;-). Note that the value of `targetprefix` has to be manually edited in the file.

```
#!/bin/sh
newroot=$1/root
targetprefix=/tuxbox/cdkroot

make console_tools

install $targetprefix/bin/loadkeys $newroot/bin
install -d $newroot/share/keymaps/i386/qwertz
install -d $newroot/share/keymaps/i386/include
install -m 444
$targetprefix/share/keymaps/i386/qwertz/de-latin1-nodeadkeys.kmap.gz
$newroot/share/keymaps/i386/qwertz
install -m 444 $targetprefix/share/keymaps/i386/qwertz/de-latin1.kmap.gz
$newroot/share/keymaps/i386/qwertz
install -m 444
$targetprefix/share/keymaps/i386/include/linux-keys-bare.inc.gz
$newroot/share/keymaps/i386/include
install -m 444
$targetprefix/share/keymaps/i386/include/linux-with-alt-and-altgr.inc.gz
$newroot/share/keymaps/i386/include
install -m 444
$targetprefix/share/keymaps/i386/include/qwertz-layout.inc.gz
$newroot/share/keymaps/i386/include
```