ckb-next

v0.2.8 at branch master

Generated by Doxygen 1.8.6

Wed May 24 2017 08:16:55

Contents

Chapter 1

ckb-next: RGB Driver for Linux and OS X

ckb-next is an open-source driver for Corsair keyboards and mice. It aims to bring the features of their proprietary CUE software to the Linux and Mac operating systems. This project is currently a work in progress, but it already supports much of the same functionality, including full RGB animations. More features are coming soon. Testing and bug reports are appreciated!

Disclaimer: ckb-next is not an official Corsair product. It is licensed under the GNU General Public License (version 2) in the hope that it will be useful, but with NO WARRANTY of any kind.

See also:

- https://github.com/mattanger/ckb-next/blob/master/DAEMON.md "Manual for the driver daemon"
- ckb testing repository (updated more frequently, but may be unstable)

Device Support

Keyboards

- K65 RGB
- K70
- K70 RGB
- K70 LUX RGB
- K95∗
- K95 RGB
- Strafe
- Strafe RGB
- = hardware playback not supported. Settings will be saved to software only.

Mice

- M65 RGB
- M65 PRO RGB
- Sabre RGB
- Scimitar RGB

Linux Installation

Pre-made packages

- Fedora 24/25, CentOS/RHEL 7 (maintained by):
  - `johanh/ckb` - based on `master` branch
- Arch Linux (maintained by ,):
  - `aur/ckb-next-git` - based on `master` branch (more stable)
  - `aur/ckb-next-latest-git` - based on `testing` branch (less stable but fresher)

If you are a package maintainer or want to discuss something with package maintainers let us know in #5, so we can have an accountable and centralized communication about this. *If you would like to maintain a package for your favorite distro/OS, please let us know as well.*

Preparation

ckb-next requires Qt5 (Qt 5.8 is recommended), libudev, zlib, gcc, g++, and glibc.

- Ubuntu: `sudo apt-get install build-essential libudev-dev qt5-default zlib1g-dev libappindicator-dev`
- Fedora: `sudo dnf install zlib-devel qt5-qtbase-devel libgudev-devel libappindicator-de systemd-devel gcc-c++`
- Arch: `sudo pacman -S base-devel qt5-base zlib`
- Other distros: Look for `qt5` or `libqt5*-devel`

Note: If you build your own kernels, ckb-next requires the `CONFIG_INPUT_UINPUT` flag to be enabled. It is located in `Device Drivers -> Input Device Support -> Miscellaneous devices -> User level driver support`. If you don't know what this means, you can ignore this.

Installing

You can download ckb-next using the "Download zip" option on the right or clone it using `git clone`. Extract it and open the ckb-master directory in a terminal. Run `./quickinstall`. It will attempt to build ckb and then ask if you'd like to install/run the application. If the build doesn't succeed, or if you'd like to hand-tune the compilation of ckb, see https://github.com/mattanger/ckb-next/blob/master/BUILD.md "'BUILD.md'" for instructions.

Upgrading

To install a new version of ckb, or to reinstall the same version, first delete the ckb-master directory and the zip file from your previous download. Then download the source code again and re-run `./quickinstall`. The script will automatically replace the previous installation. You may need to reboot afterward.

Uninstalling

First, stop the ckb-daemon service and remove the service file.

- If you have systemd (Ubuntu versions starting with 15.04): "' sudo systemctl stop ckb-daemon sudo rm -f /usr/lib/systemd/system/ckb-daemon.service "'
- If you have Upstart (Ubuntu versions earlier than 15.04): "' sudo service ckb-daemon stop sudo rm -f /etc/init/ckb-daemon.conf "'
- If you have OpenRC: "' sudo rc-service ckb-daemon stop sudo rc-update del ckb-daemon default sudo rm -f /etc/init.d/ckb-daemon "
- `If you're not sure, re-run the`quickinstall`script and proceed to the service installation. The script will say`System service: Upstart detected`or`System service: systemd detected'. Please be aware that OpenRC is currently not detected automatically.

Afterward, remove the applications and related files: "' sudo rm -f /usr/bin/ckb /usr/bin/ckb-daemon /usr/share/applications/ckb.desktop /usr/share/icons/hicolor/512x512/apps/ckb.png sudo rm -rf /usr/lib/ckb-animations "'

Before `https://github.com/mattanger/ckb-next/commit/f347e60df211c60452f95084b6c46dc4ec5f42` animations were located elsewhere, try removing them as well: "' sudo rm -rf /usr/bin/ckb-animations "'

OS X/macOS Installation

Binary download

macOS `pkg` can be downloaded from `GitHub Releases`. It is always built with the last available stable Qt version and tagrets 10.10 SDK. If you run 10.9.x, you'll need to build the project from source and comment out `src/ckb-heat` (and the backslash above it) inside `ckb.pro`.

Building from source

Install the latest version of Xcode from the App Store. While it's downloading, open the Terminal and execute `xcode-select --install` to install Command Line Tools. Then open Xcode, accept the license agreement and wait for it to install any additional components (if necessary). When you see the "Welcome to Xcode" screen, from the top bar choose `Xcode -> Preferences -> Locations -> Command Line Tools` and select an SDK version. Afterwards install Qt5 from here: `http://www.qt.io/download-open-source/`

The easiest way to build the driver is with the `quickinstall` script, which is present in the ckb-master folder. Double-click on `quickinstall` and it will compile the app for you, then ask if you'd like to install it system-wide. If the build fails for any reason, or if you'd like to compile and install manually, see https://github.com/ccMS-C/ckb/blob/master/BUILD.md "'BUILD.md'".

Upgrading (binary)

Download the latest `ckb.pkg`, run the installer, and reboot. The newly-installed driver will replace the old one.

Upgrading (source)

Remove the existing ckb-master directory and zip file. Re-download the source code and run the `quickinstall` script again. The script will automatically replace the previous installation. You may need to reboot afterward.

Uninstalling

Drag `ckb.app` into the trash. Then stop and remove the agent:

"'sh sudo unload /Library/LaunchDaemons/com.ckb.daemon.plist sudo rm /Library/LaunchDaemons/com.ckb.-daemon.plist "'

Usage

The user interface is still a work in progress.

Major features

- Control multiple devices independently
- United States and European keyboard layouts
- Customizable key bindings
- Per-key lighting and animation
- Reactive lighting
- Multiple profiles/modes with hardware save function
- Adjustable mouse DPI with ability to change DPI on button press

Closing ckb will actually minimize it to the system tray. Use the Quit option from the tray icon or the settings screen to exit the application.

Roadmap

- **v0.3 release:**
- Ability to store profiles separately from devices, import/export them
- More functions for the Win Lock key
- Key macros
- **v0.4 release:**
- Ability to import CUE profiles
- Ability to tie profiles to which application has focus
- **v0.5 release:**
- Key combos
- Timers?
- **v1.0 release:**
- OSD? (Not sure if this can actually be done)
- Extra settings?
- ????

Troubleshooting

Linux

If you have problems connecting the device to your system (device doesn't respond, ckb-daemon doesn't recognize or can't connect it) and/or you experience long boot times when using the keyboard, try adding the following to your kernel's `cmdline`:

- K65 RGB: `usbhid.quirks=0x1B1C:0x1B17:0x20000408`
- K70: `usbhid.quirks=0x1B1C:0x1B09:0x20000408`
- K70 LUX: `usbhid.quirks=0x1B1C:0x1B36:0x20000408`
- K70 RGB: `usbhid.quirks=0x1B1C:0x1B13:0x20000408`
- K95: `usbhid.quirks=0x1B1C:0x1B08:0x20000408`
- K95 RGB: `usbhid.quirks=0x1B1C:0x1B11:0x20000408`
- Strafe: `usbhid.quirks=0x1B1C:0x1B15:0x20000408`
- Strafe RGB: `usbhid.quirks=0x1B1C:0x1B20:0x20000408`
- M65 RGB: `usbhid.quirks=0x1B1C:0x1B12:0x20000408`
- Sabre RGB Optical: `usbhid.quirks=0x1B1C:0x1B14:0x20000408`
- Sabre RGB Laser: `usbhid.quirks=0x1B1C:0x1B19:0x20000408`
- Scimitar RGB: `usbhid.quirks=0x1B1C:0x1B1E:0x20000408`

For instructions on adding `cmdline` parameters in Ubuntu, see https://wiki.ubuntu.com/Kernel/-KernelBootParameters

If you have multiple devices, combine them with commas, starting after the =. For instance, for K70 RGB + M65 RGB: `usbhid.quirks=0x1B1C:0x1B13:0x20000408,0x1B1C:0x1B12:0x20000408`

If it still doesn't work, try replacing `0x20000408` with `0x4`. Note that this will cause the kernel driver to ignore the device(s) completely, so you need to ensure ckb-daemon is running at boot or else you'll have no input. This will not work if you are using full-disk encryption.

If you see **GLib** critical errors like "' GLib-GObject-CRITICAL **: g_type_add_interface_static: assertion 'G_TYPE-_IS_INSTANTIATABLE (instance_type)' failed "' read `this Arch Linux thread` and try different combinations from it. If it doesn't help, you might want get support from your distribution community and tell them you cannot solve the problem in this thread.

If you're using **Unity** and the tray icon doesn't appear correctly, run `sudo apt-get install libappindicator-dev`. Then reinstall ckb.

OS X/macOS

- **"ckb.pkg" can't be opened because it is from an unidentified developer** Open `System Preferences > Security & Privacy > General` and click `Open Anyway`.
- **Modifier keys (Shift, Ctrl, etc.) are not rebound correctly** ckb does not recognize modifier keys rebound from System Preferences. You can rebind them again within the application.
- **~ key prints §±** Check your keyboard layout on ckb's Settings screen. Choose the layout that matches your physical keyboard.
- **Compile problems** Can usually be resolved by rebooting your computer and/or reinstalling Qt. Make sure that Xcode works on its own. If a compile fails, delete the `ckb-master` directory as well as any automatically generated `build-ckb` folders and try again from a new download.
- **Scroll wheel does not scroll** As of #c3474d2 it's now possible to **disable scroll acceleration** from the GUI. You can access it under "OSX tweaks" in the "More settings" screen. Once disabled, the scroll wheel should behave consistently.

General

Please ensure your keyboard firmware is up to date. If you've just bought the keyboard, connect it to a Windows computer first and update the firmware from Corsair's official utility.

Before reporting an issue, connect your keyboard to a Windows computer and see if the problem still occurs. If it does, contact Corsair. Additionally, please check the Corsair user forums to see if your issue has been reported by other users. If so, try their solutions first.

Common issues:

- **Problem:** ckb says "No devices connected" or "Driver inactive"
- **Solution:** Try rebooting the computer and/or reinstalling ckb. Try removing the keyboard and plugging it back in. If the error doesn't go away, try the following:
- **Problem:** Keyboard doesn't work in BIOS, doesn't work at boot
- **Solution:** Some BIOSes have trouble communicating with the keyboard. They may prevent the keyboard from working correctly in the operating system as well. First, try booting the OS *without* the keyboard attached, and plug the keyboard in after logging in. If the keyboard works after the computer is running but does not work at boot, you may need to use the keyboard's BIOS mode option.
- BIOS mode can be activated using the poll rate switch at the back of the keyboard. Slide it all the way to the position marked "BIOS". You should see the scroll lock light blinking to indicate that it is on. (Note: Unfortunately, this has its own problems - see Known Issues. You may need to activate BIOS mode when booting the computer and deactivate it after logging in).

- **Problem:** Keyboard isn't detected when plugged in, even if driver is already running
- **Solution:** Try moving to a different USB port. Be sure to follow Corsair's USB connection requirements. Note that the keyboard does not work with some USB3 controllers - if you have problems with USB3 ports, try USB2 instead. If you have any USB hubs on hand, try those as well. You may also have success sliding the poll switch back and forth a few times.

Reporting issues

If you have a problem that you can't solve (and it isn't mentioned in the Known Issues section below), you can report it on the GitHub issue tracker. Before opening a new issue, please check to see if someone else has reported your problem already - if so, feel free to leave a comment there.

Known issues

- Using the keyboard in BIOS mode prevents the media keys (including mute and volume wheel), as well as the K95's G-keys from working. This is a hardware limitation.
- The tray icon doesn't appear in some desktop environments. This is a known Qt bug. If you can't see the icon, reopen ckb to bring the window back.
- When starting the driver manually, the Terminal window sometimes gets spammed with enter keys. You can stop it by unplugging and replugging the keyboard or by moving the poll rate switch.
- When stopping the driver manually, the keyboard sometimes stops working completely. You can reconnect it by moving the poll rate switch.
- On newer versions of macOS (i.e. 10.12 and up) CMD/Shift+select does not work, yet. Stopping the daemon and GUI for `ckb` will fix this issue temporarily.

Contributing

You can contribute to the project by opening a pull request. It's best if you base your changes off of the `testing` branch as opposed to the `master`, because the pull request will be merged there first. If you'd like to contribute but don't know what you can do, take a look at the issue tracker and see if any features/problems are still unresolved. Feel free to ask if you'd like some ideas.

Contact us

There are multiple ways you can get in touch with us:

- join `ckb-next` mailing list
- open a GitHub Issue
- hop on `#ckb-next` to chat

What happened to the original ckb

Due to time restrictions, the original author of **ckb** ccMSC hasn't been able to further develop the software. So the community around it decided to take the project over and continue its development. That's how **ckb-next** was created. Currently it's not rock solid and not very easy to set up on newer systems but we are actively working on this. Nevertheless the project already incorporates a notable amount of fixes and patches in comparison to the original ckb.

Chapter 2

Building ckb

Linux

You can build the project by running `./qmake-auto && make` in a Terminal inside the ckb-master directory. The binaries will be placed in a new `bin` directory assuming they compile successfully. If you get a `No suitable qmake found` error, make sure Qt5 is installed and up to date. You may have to invoke qmake manually, then run `make` on its own. If you have Qt Creator installed, you can open `ckb.pro` (when asked to configure the project, make sure "Desktop" is checked) and use `Build > Build Project "ckb" (Ctrl+B)` to build the application instead.

Running as a service:

First copy the binary and the service files to their system directories:

- Upstart (Ubuntu, prior to 15.04): `sudo cp -R bin/* /usr/bin && sudo cp service/upstart/ckb-daemon.conf /etc/init`
- Systemd (Ubuntu 15.04 and later): `sudo cp -R bin/* /usr/bin && sudo cp service/systemd/ckb-daemon.service /usr/lib/systemd/system`
- OpenRC: `sudo cp -R bin/* /usr/bin && sudo cp service/openrc/ckb-daemon /etc/init.d/`

To launch the driver and enable it at start-up:

- Upstart: `sudo service ckb-daemon start`
- Systemd: `sudo systemctl start ckb-daemon && sudo systemctl enable ckb-daemon`
- OpenRC: `sudo rc-service ckb-daemon start && sudo rc-update add ckb-daemon default`

Open the `bin` directory and double-click on `ckb` to launch the user interface. If you want to run it at login, add `ckb --background` to your Startup Applications.

Running manually:

Open the `bin` directory in a Terminal and run `sudo ./ckb-daemon` to start the driver. To start the user interface, run `./ckb`. Running the driver manually may be useful for testing/debugging purposes, but you must leave the terminal window open and you'll have to re-run it at every reboot, so installing it as a service is the best long-term solution.

OSX

Open ckb.pro in Qt Creator. You should be prompted to configure the project (make sure the "Desktop" configuration is selected and not iOS). Once it's finished loading, press `Cmd+B` or select `Build > Build Project "ckb"` from the menu bar. When it's done, you should see a newly-created `ckb.app` in the project directory. Exit Qt Creator.

Alternatively, open a Terminal in the ckb-master directory and run `./qmake-auto && make`. It will detect Qt automatically if you installed it to one of the standard locations. You should see a newly created `ckb.app` if the build is successful.

Running as a service:

Copy `ckb.app` to your Applications folder. Copy the file `'service/launchd/com.ckb.daemon.plist'` to your computer's `/Library/LaunchDaemons` folder (you can get to it by pressing `Cmd+Shift+G` in Finder and typing the location). Then open a Terminal and run the following commands to launch the driver:

"' sudo chown root:wheel /Library/LaunchDaemons/com.ckb.daemon.plist sudo chmod 0700 /Library/LaunchDaemons/com.ckb.daemon.plist sudo launchctl load /Library/LaunchDaemons/com.ckb.daemon.plist "'

After you're done, open `ckb.app` to launch the user interface.

Running manually:

Open a Terminal in the ckb directory and run `sudo ckb.app/Contents/Resources/ckb-daemon` to start the driver. Open `ckb.app` to start the user interface. Note that you must leave the terminal window open and must re-launch the driver at every boot if you choose this; installing as a service is the better long term solution.

Chapter 3

cbk Improvements Roadmap

Short term plan

- merge existing PR submitted to original ckb repo
- Contact other developers interested in collaboration on a new and improved version of ckb
- Figure out the issues relating to MacOS Sierra and other version
- Device support:
  - Determine which will need support other than just USB id additions
- Address existing bugs. Not help requests.

Chapter 4

DAEMON

The daemon provides devices at `/dev/input/ckb*`, where `*` is the device number, starting at 1. Up to 9 devices may be connected at once and controlled independently. The daemon additionally provides `/dev/input/ckb0`, which stores driver information.

Mac note: The devices on OSX are located at `/var/run/ckb*` and not `/dev/input/ckb*`. So wherever you see `/dev/input` in this document, replace it with `/var/run`.

`/dev/input/ckb0` contains the following files:

- `connected`: A list of all connected devices, one per line. Each line contains a device path followed by the device's serial number and its description.
- `pid`: The process identifier of the daemon.
- `version`: The daemon version.

Other `ckb*` devices contain the following:

- `cmd`: Keyboard controller.
- `notify0`: Keyboard notifications.
- `features`: Device features.
- `fwversion`: Device firmware version (not present on all devices).
- `model`: Device description/model.
- `pollrate`: Poll rate in milliseconds (not present on all devices).
- `serial`: Device serial number. `model` and `serial` will match the info found in `ckb0/connected`

Commands

The `/dev/input/ckb*/cmd` nodes accept input in the form of text commands. They are normally accessible to all users on the system (see Security section). Commands should be given in the following format: `[mode <n>] command1 [paramter1] [command2] [parameter2] [command3] [parameter3] ...`

In a terminal shell, you can do this like `echo mycommand > /dev/input/ckb1/cmd`. Programmatically, you can open and write them as regular files. When programming, you must append a newline character and flush the output before your command(s) will actually be read.

The `mode` parameter is used to group settings. Most (but not all) settings are mode-specific; that is, changing mode 1 will not affect mode 2. By default, all commands affect the current mode. Use `mode <n> switch` to change the current mode.

When plugged in, all devices start in hardware-controlled mode (also known as idle mode) and will not respond to commands. Before issuing any other commands, write `active` to the command node, like `echo active > /dev/input/ckb1/cmd`. To put the device back into hardware mode, issue the `idle` command.

Features

The `features` node describes features supported by the device, which may not be present on all devices. The first two words in the `features` node are always `<vendor> <model>`, like `corsair k70`. After that, any of the following features may appear:

- `adjrate`: Device supports adjustable poll rate.
- `bind`: Device supports key rebinding.
- `fwupdate`: Device supports firmware updates.
- `fwversion`: Device has a detectable firmware version (stored in the `fwversion` node).
- `notify`: Device supports key notifications.
- `pollrate`: Device has a detectable poll rate (stored in the `pollrate` node).
- `rgb`: Device supports RGB lighting.

Keyboard layout

The driver has no concept of keyboard layouts; all keys are referred to by their English names regardless of the underlying hardware. This means that, for instance, in an AZERTY layout the `q` key in ckb-daemon corresponds to A on the physical keyboard. Note that on UK/european (ISO) layouts, the backslash key (beside left shift) is called `bslash_iso`, while `bslash` refers to the backslash on the US keyboard. The key next to Enter on the ISO keyboard is known as `hash`. See src/ckb-daemon/keymap.c for the full table of supported keys.

For technical reasons, the OSX driver may swap the `bslash_iso` and `grave` keys if the keyboard layout is not set correctly. To compensate for this, write `layout iso` or `layout ansi` to the command node.

Poll rate

A device's current poll rate can be read from its `pollrate` node, assuming it has one. Keyboards have a hardware switch to control poll rate and cannot be adjusted via software. However, mice have a software-controlled poll rate. You can change it by issuing `pollrate <interval>` to the command node, where `interval` is the time in milliseconds. Valid poll rates are `1`, `2`, `4`, and `8`.

Profiles and modes

Each mode has its own independent binding and lighting setup. When the daemon starts or a keyboard is plugged in, the profile will be loaded from the hardware. By default, all commands will update the currently selected mode. The `mode <n>` command may be used to change the settings for a different mode. Up to 6 modes are available. Each keyboard has one profile, which may be given a name. Mode 1 may be saved to the device hardware, or modes 1-3 in the case of the K95. Modes 4 through 6 are software-only. Profile management commands are as follows:

- `profilename <name>` sets the profile's name. The name must be written without spaces; to add a space, use `%20`.
- `name <name>` sets the current mode's name. Use `mode <n> name <name>` to set a different mode's name.

- `profileid <guid> [<modification>]` sets a profile's ID. The GUID must be written in registry format, like `{12345678-ABCD-EF01-2345-6789ABCDEF01}`. The optional modification number must be written with 8 hex digits, like `ABCDEF01`.
- `id <guid> [<modification>]` sets a mode's ID.
- `mode <n> switch` switches the keyboard to mode N. If the mode does not exist, it will be created with a blank ID, black lighting, and default bindings.
- `hwload` loads the RGB profile from the hardware. Key bindings and non-hardware RGB modes are unaffected.
- `hwsave` saves the RGB profile to the hardware.
- `erase` erases the current mode, resetting its lighting and bindings. Use `mode <n> erase` to erase a different mode.
- `eraseprofile` erases the entire profile, deleting its name, ID, and all of its modes.

Examples:

- `profilename My%20Profile mode 1 name Mode%201 mode 2 name Mode%202 mode 3 name Mode%203` will name the profile "My Profile" and name modes 1-3 "Mode 1", "Mode 2", and "Mode 3".
- `eraseprofile hwload` resets the entire profile to its hardware settings.

LED commands

The backlighting is controlled by the `rgb` commands.

- `rgb <RRGGBB>` sets the entire keyboard to the color specified by the hex constant RRGGBB.
- `rgb <key>:<RRGGBB>` sets the specified key to the specified hex color.

Examples:

- `rgb ffffff` makes the whole keyboard white.
- `rgb 000000` makes the whole keyboard black.
- `rgb esc:ff0000` sets the Esc key red but leaves the rest of the keyboard unchanged.

Multiple keys may be changed to one color when separated with commas, for instance:

- `rgb w,a,s,d:0000ff` sets the WASD keys to blue.

Additionally, multiple commands may be combined into one, for instance:

- `rgb ffffff esc:ff0000 w,a,s,d:0000ff` sets the Esc key red, the WASD keys blue, and the rest of the keyboard white (note the lack of a key name before `ffffff`, implying the whole keyboard is to be set).

By default, the controller runs at 30 FPS, meaning that attempts to animate the LEDs faster than that will be ignored. If you wish to change it, send the command `fps <n>`. The maximum frame rate is 60.

For devices running in 512-color mode, color dithering can be enabled by sending the command `dither 1`. The command `dither 0` disables dithering.

Indicators

The indicator LEDs (Num Lock, Caps Lock, Scroll Lock) are controlled with the `i` commands.

- `ioff <led>` turns an indicator off permanently. Valid LED names are `num`, `caps`, and `scroll`.
- `ion <led>` turns an indicator on permanently.
- `iauto <led>` turns an indicator off or on automatically (default behavior).

Binding keys

Keys may be rebound through use of the `bind` commands. Binding is a 1-to-1 operation that translates one keypress to a different keypress regardless of circumstance.

- `bind <key1>:<key2>` remaps key1 to key2.
- `unbind <key>` unbinds a key, causing it to lose all function.
- `rebind <key>` resets a key, returning it to its default binding.

Examples:

- `bind g1:esc` makes G1 become an alternate Esc key (the actual Esc key is not changed).
- `bind caps:tab tab:caps` switches the functions of the Tab and Caps Lock keys.
- `unbind lwin rwin` disables both Windows keys, even without using the keyboard's Windows Lock function.
- `rebind all` resets the whole keyboard to its default bindings.

Key macros

Macros are a more advanced form of key binding, controlled with the `macro` command.

- `macro <keys>:<command>` binds a key combination to a command, where the command is a series of key presses. To combine keys, separate them with `+`; for instance, `lctrl+a` binds a macro to (left) Ctrl+A. In the command field, enter `+<key>` to trigger a key down or `-<key>` to trigger a key up. To simulate a key press, use `+<key>,-<key>`.
- `macro <keys>:clear` clears commands associated with a key combination. Only one macro may be assigned per combination; assigning a second one will overwrite the first.
- `macro clear` clears all macros.

Examples:

- `macro g1:+lctrl,+a,-a,-lctrl` triggers a Ctrl+A when G1 is pressed.
- `macro g2+g3:+lalt,+f4,-f4,-lalt` triggers an Alt+F4 when G2 and G3 are pressed simultaneously.

Assigning a macro to a key will cause its binding to be ignored; for instance, `macro a:+b,-b` will cause A to generate a B character regardless of its binding. However, `macro lctrl+a:+b,-b` will cause A to generate a B only when Ctrl is also held down.

DPI and mouse settings

DPI settings are stored in a bank. They are controlled with the `dpi` command.

- `dpi <stage>:<x>,<y>` sets the DPI for a given `stage` to `x` by `y`. Valid stages are `0` through `5`. In hardware, `1` is the first (lowest) stage and `5` is the highest. Stage `0` is used for Sniper mode.
- `dpi <stage>:<xy>` sets both X and Y.
- `dpi <stage>:off` disables a DPI stage.
- `dpisel <stage>` sets the current stage selection.

In order to change the mouse's current DPI, first update one of the stages with the value you want, then select that stage. For instance:

- `dpi 1:1000 dpisel 1` sets the current DPI to 1000x1000.

Additional mouse settings:

- `lift <height>` sets the lift height, from `1` (lowest) to `5` (highest)
- `snap <on|off>` enables or disables Angle Snap.

Notifications

The keyboard can be configured to generate user-readable notifications on keypress events. These are controlled with the `notify` commands. In order to see events, read from `/dev/input/ckb*/notify0`. In a terminal, you can do this like `cat /dev/input/ckb1/notify0`. Programmatically, you can open it for reading like a regular file.

Note that the file can only reliably be read by one application: if you try to open it in two different programs, they may both fail to get data. Data will be buffered as long as no programs are reading, so you will receive all unread notifications as soon as you open the file. If you'd like to read notifications from two separate applications, send the command `notifyon <n>` to the keyboard you wish to receive notifications from, where N is a number between 1 and 9. If `/dev/input/ckb*/notify<n>` does not already exist, it will be created, and you can read notifications from there without disrupting any other program. To close a notification node, send `notifyoff <n>`.

`notify0` is always open and will not be affected by `notifyon`/`notifyoff` commands. By default, all notifications are printed to `notify0`. To print output to a different node, prefix your command with `@<node>`.

Notifications are printed with one notification per line. Commands are as follows:

- `notify <key>:on` or simply `notify <key>` enables notifications for a key. Each key will generate two notifications: `key +<key>` when the key is pressed, and `key -<key>` when it is released.
- `notify <key>:off` turns notifications off for a key.

Examples:

- `notify w a s d` sends notifications whenever W, A, S, or D is pressed.
- `notify g1 g2 g3 g4 g5 g6 g7 g8 g9 g10 g11 g12 g13 g14 g15 g16 g17 g18 mr m1 m2 m3 light lock` prints a notification whenever a non-standard key is pressed.
- `notify all:off` turns all key notifications off.
- `@5 notify esc` prints Esc key notifications to `notify5`.

Note: Key notifications are *not* affected by bindings. For instance, if you run `echo bind a:b notify a > /dev/input/ckb1/cmd` and then press the A key, the notifications will read `key +a key -a`, despite the fact that the character printed on screen will be `b`. Likewise, unbinding a key or assigning a macro to a key does not affect the notifications.

Indicator notifications

You can also choose to receive notifications for the indicator LEDs by using the `inotify` command. For instance, `inotify caps:on` or simply `inotify caps` will print notifications whenever the Caps Lock LED is toggled. The notifications will read `i +caps` when the light is turned on and `i -caps` when it is turned off. It is also possible to toggle all indicators at once using `inotify all` or `inotify all:off`.

Like key notifications, indicator notifications are not affected by bindings, nor by the `ion`, `ioff`, or `iauto` commands. The notifications will reflect the state of the LEDs as seen be the event device.

Getting parameters

Parameters can be retrieved using the `get` command. The data will be sent out as a notification. Generally, the syntax to get the data associated with a command is `get :<command>` (note the colon), and the associated data will be returned in the form of `<command> <data>`. The following data may be gotten:

- `get :mode` returns the current mode in the form of a `switch` command. (Note: Do not use this in a line containing a `mode` command or it will return the mode that you selected, rather than the keyboard's current mode.)
- `get :name` returns the current mode's name in the form of `mode <n> name <name>`. To see the name of another mode, use `mode <n> get :name`. The name is URL-encoded; spaces are written as %20. The name may be truncated, so `name <some long string> get :name` may return something shorter than what was entered.
- `get :profilename` returns the profile's name, in the form of `profilename <name>`. As above, it is URL-encoded and may be truncated.
- `get :hwname` and `get :hwprofilename` return the same thing except taken from the current hardware profile instead of the in-memory profile. The output is identical but will read `hwname` instead of `name` and `hwprofilename` instead of `profilename`.
- `get :id` returns the current mode's ID and modification number in the form of `mode <n> id <guid> <modification>`.
- `get :profileid` returns the current profile's ID and modification number in the form of `profileid <guid> <modification>`.
- `get :hwid` and `get :hwprofileid` return the same thing except from the current hardware profile/mode. As before, the ouput will be the same but with `hwid` and `hwprofileid` instead of `id` and `profileid`.
- `get :rgb` returns an `rgb` command equivalent to the current RGB state.
- `get :hwrgb` does the same thing, but retrieves the colors currently stored in the hardware profile. The output will say `hwrgb` instead of `rgb`.
- `get :dpi` returns a `dpi` command equivalent to the current DPI bank.
- `get :dpisel` returns a `dpisel` command for the currently-selected DPI stage.
- `get :lift` returns a `lift` command for the current lift height.
- `get :snap` returns the current angle snap status.
- `get :hwdpi`, `get :hwdpisel`, `get :hwlift`, and `get :hwsnap` return the same properties, but for the current hardware profile.
- `get :keys` and `get :i` return the current keypress status and indicator status, respectively. They will indicate all currently pressed keys and all currently active indicators, like `key +enter` and `i +num`.

Like `notify`, you must prefix your command with `@<node>` to get data printed to a node other than `notify0`.

Firmware updates

WARNING: Improper use of `fwupdate` may brick your device; use this command *at your own risk*. I accept no responsibility for broken keyboards.

The latest firmware versions and their URLs can be found in the `FIRMWARE` document. To update your keyboard's firmware, first extract the contents of the zip file and then issue the command `fwupdate /path/to/fw/file.bin` to the keyboard you wish to update. The path name must be absolute and must not include spaces. If it succeeded, you should see `fwupdate <path> ok` logged to the keyboard's notification node and then the device will disconnect and reconnect. If you see `fwupdate <path> invalid` it means that the firmware file was not valid for the device; more info may be available in the daemon's `stdout`. If you see `fwupdate <path> fail` it means that the file was valid but the update failed at a hardware level. The keyboard may disconnect/reconnect anyway or it may remain in operation.

When the device reconnects you should see the new firmware version in its `fwversion` node; if you see `0000` instead it means that the keyboard did not update successfully and will need another `fwupdate` command in order to function again. If the update fails repeatedly, try connecting the keyboard to a Windows PC and using the official firmware update in CUE.

Restart

Because sometimes the communication between the daemon and the keyboard is corrupted after resuming from standby or suspend, a restart function is implemented. It first calls the quit() funtion, then it calls main() again with the original parameter list.

There are two ways to restart the daemon:

- send the string "restart some-description-as-one-word" to the cmd-pipe (normally /dev/input/ckb1/cmd or /dev/input/ckb2/cmd, depending on what device gets which ID.
- send SIGUSR1 to the daemon process (as root).

Later on, there may be a user interface in the client for the first method.

Security

By default, all of the `ckb*` nodes may be accessed by any user. For most single-user systems this should not present any security issues, since only one person will have access to the computer anyway. However, if you'd like to restrict the users that can write to the `cmd` nodes or read from the `notify` nodes, you can specify the `--gid=<group>` option at start up. For instance, on most systems you could run `ckb-daemon --gid=1000` to make them accessible only by the system's primary user. `ckb-daemon` must still be run as root, regardless of which `gid` you specify. The `gid` option may be set only at startup and cannot be changed while the daemon is running.

The daemon additionally supports a `--nonotify` option to disable key notifications, to prevent unauthorized programs from logging key input. Note that this will interfere with some of `ckb`'s abilities. It is also highly unlikely to increase security unless you are using the program in a stripped down terminal environment without Xorg. For most use cases there are many other (more likely) ways that a keylogger program could compromise your system. Nevertheless, the option is provided for the sake of paranoia. If you'd like to disable key rebinding as well, launch the daemon with `--nobind`. `--nobind` implies `--nonotify`, so notifications will also be disabled. As with `--gid`, these options must be set at startup and cannot be changed while the daemon is running.

Chapter 5

Data Structure Index

5.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 6

File Index

6.1 File List

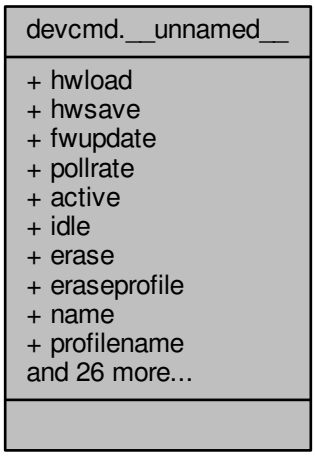
Here is a list of all files with brief descriptions:

Chapter 7

Data Structure Documentation

7.1 devcmd.__unnamed__ Struct Reference

Collaboration diagram for devcmd.__unnamed__:



Data Fields

- cmdhandler_io hwload
- cmdhandler_io hwsave
- cmdhandler_io fwupdate
- cmdhandler_io pollrate
- cmdhandler_io active
- cmdhandler_io idle
- cmdhandler erase
- cmdhandler eraseprofile
- cmdhandler name
- cmdhandler profilename
- cmdhandler id

- cmdhandler profileid
- cmdhandler rgb
- cmdhandler ioff
- cmdhandler ion
- cmdhandler iauto
- cmdhandler bind
- cmdhandler unbind
- cmdhandler rebind
- cmdhandler_mac macro
- cmdhandler_mac dpi
- cmdhandler dpisel
- cmdhandler lift
- cmdhandler snap
- cmdhandler notify
- cmdhandler inotify
- cmdhandler get
- cmdhandler restart
- int(∗ start)(usbdevice ∗kb, int makeactive)
- void(∗ setmodeindex)(usbdevice ∗kb, int index)
- void(∗ allocprofile)(usbdevice ∗kb)
- int(∗ loadprofile)(usbdevice ∗kb)
- void(∗ freeprofile)(usbdevice ∗kb)
- int(∗ updatergb)(usbdevice ∗kb, int force)
- void(∗ updateindicators)(usbdevice ∗kb, int force)
- int(∗ updatedpi)(usbdevice ∗kb, int force)

7.1.1 Detailed Description

Definition at line 78 of file command.h.

7.1.2 Field Documentation

7.1.2.1

7.1.2.2

7.1.2.3

7.1.2.4

7.1.2.5

7.1.2.6

7.1.2.7

7.1.2.8

7.1.2.9

7.1.2.10

7.1.2.11

7.1.2.12

7.1.2.13

7.1.2.14

7.1.2.15

7.1.2.16

7.1.2.17

7.1.2.18

7.1.2.19

7.1.2.20

7.1.2.21

7.1.2.22

7.1.2.23

7.1.2.24

7.1.2.25

7.1.2.26

7.1.2.27

7.1.2.28

7.1.2.29

7.1.2.30

7.1.2.31

7.1.2.32

7.1.2.33

7.1.2.34

7.1.2.35

7.1.2.36

The documentation for this struct was generated from the following files:

Chapter 8

File Documentation

8.1 BUILD.md File Reference

8.2 DAEMON.md File Reference

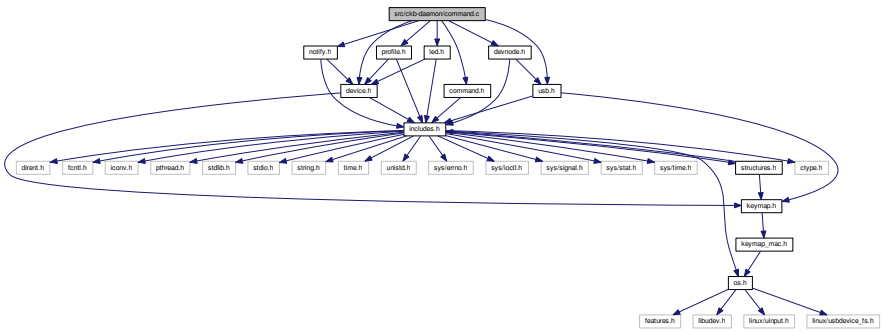
8.3 README.md File Reference

8.4 ROADMAP.md File Reference

8.5 src/ckb-daemon/command.c File Reference

```
#include "command.h"
#include "device.h"
#include "devnode.h"
#include "led.h"
#include "notify.h"
#include "profile.h"
#include "usb.h"
```

Include dependency graph for command.c:



Macros

- #define TRY_WITH_RESET(action)

Functions

- int readcmd (usbdevice *kb, const char *line)

Variables

- static const char *const cmd_strings [(CMD_LAST-CMD_FIRST+2)-1]

8.5.1 Macro Definition Documentation

8.5.1.1 #define TRY_WITH_RESET(*action*)

Value:

```
while(action){              \
        if(usb_tryreset(kb)){   \
            free(word);         \
            return 1;           \
        }                       \
    }
```

Definition at line 58 of file command.c.

Referenced by readcmd().

8.5.2 Function Documentation

8.5.2.1 int readcmd (usbdevice * *kb*, const char * *line*)

Because length of word is length of line + 1, there should be no problem with buffer overflow.

Definition at line 67 of file command.c.

References ACCEL, ACTIVE, usbdevice::active, BIND, CMD_COUNT, CMD_FIRST, cmd_strings, usbprofile::currentmode, DELAY, usbdevice::delay, DITHER, usbdevice::dither, devcmd::do_cmd, devcmd::do_io, devcmd::do_macro, DPI, DPISEL, ERASE, ERASEPROFILE, FEAT_ANSI, FEAT_BIND, FEAT_ISO, FEAT_LMASK, FEAT_MOUSEACCEL, FEAT_NOTIFY, usbdevice::features, lighting::forceupdate, FPS, FWUPDATE, GET, HAS_FEATURES, HWLOAD, HWSAVE, IAUTO, ID, IDLE, INDEX_OF, INOTIFY, IOFF, ION, IS_FULLRANGE, IS_MOUSE_DEV, keymap, LAYOUT, LIFT, usbmode::light, MACRO, mknotifynode(), MODE, usbprofile::mode, MODE_COUNT, N_KEYS_EXTENDED, NAME, NEEDS_FW_UPDATE, NONE, NOTIFY, NOTIFYOFF, NOTIFYON, OUTFIFO_MAX, POLLRATE, usbdevice::profile, PROFILEID, PROFILENAME, REBIND, RESTART, RGB, rmnotifynode(), SCROLL_ACCELERATED, SCROLL_MAX, SCROLL_MIN, SCROLLSPEED, SNAP, SWITCH, TRY_WITH_RESET, UNBIND, usbdevice::usbdelay, and usbdevice::vtable.

Referenced by devmain().

```
67                                          {
68     char* word = malloc(strlen(line) + 1);
69     int wordlen;
70     const char* newline = 0;
71     const devcmd* vt = kb->vtable;
72     usbprofile* profile = kb->profile;
73     usbmode* mode = 0;
74     int notifynumber = 0;
75     // Read words from the input
76     cmd command = NONE;
77     while(sscanf(line, "%s%n", word, &wordlen) == 1){
78         line += wordlen;
79         // If we passed a newline, reset the context
80         if(line > newline){
81             mode = profile->currentmode;
82             command = NONE;
83             notifynumber = 0;
84             newline = strchr(line, '\n');
```

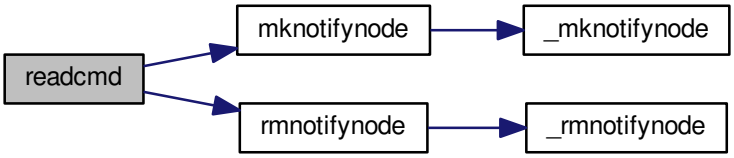
```
85              if(!newline)
86                  newline = line + strlen(line);
87          }
88          // Check for a command word
89          for(int i = 0; i < CMD_COUNT - 1; i++){
90              if(!strcmp(word, cmd_strings[i])){
91                  command = i + CMD_FIRST;
92  #ifndef OS_MAC
93                  // Layout and mouse acceleration aren't used on Linux; ignore
94                  if(command == LAYOUT || command == ACCEL || command ==
      SCROLLSPEED)
95                      command = NONE;
96  #endif
97                  // Most commands require parameters, but a few are actions in and of themselves
98                  if(command != SWITCH
99                          && command != HWLOAD && command != HWSAVE
100                          && command != ACTIVE && command != IDLE
101                          && command != ERASE && command != ERASEPROFILE
102                          && command != RESTART)
103                      goto next_loop;
104                  break;
105              }
106          }
107
108          // Set current notification node when given @number
109          int newnotify;
110          if(sscanf(word, "@%u", &newnotify) == 1 && newnotify < OUTFIFO_MAX){
111              notifynumber = newnotify;
112              continue;
113          }
114
115          // Reject unrecognized commands. Reject bind or notify related commands if the keyboard doesn't
       have the feature enabled.
116          if(command == NONE
117                  || ((!HAS_FEATURES(kb, FEAT_BIND) && (command ==
      BIND || command == UNBIND || command == REBIND || command ==
      MACRO || command == DELAY))
118                          || (!HAS_FEATURES(kb, FEAT_NOTIFY) && command ==
      NOTIFY))){
119              next_loop:
120              continue;
121          }
122          // Reject anything not related to fwupdate if device has a bricked FW
123          if(NEEDS_FW_UPDATE(kb) && command != FWUPDATE && command !=
      NOTIFYON && command != NOTIFYOFF)
124              continue;
125
126          // Specially handled commands - these are available even when keyboard is IDLE
127          switch(command){
128          case NOTIFYON: {
129              // Notification node on
130              int notify;
131              if(sscanf(word, "%u", &notify) == 1)
132                  mknotifynode(kb, notify);
133              continue;
134          } case NOTIFYOFF: {
135              // Notification node off
136              int notify;
137              if(sscanf(word, "%u", &notify) == 1 && notify != 0) // notify0 can't be removed
138                  rmnotifynode(kb, notify);
139              continue;
140          } case GET:
141              // Output data to notification node
142              vt->get(kb, mode, notifynumber, 0, word);
143              continue;
144          case LAYOUT:
145              // OSX: switch ANSI/ISO keyboard layout
146              if(!strcmp(word, "ansi"))
147                  kb->features = (kb->features & ~FEAT_LMASK) |
      FEAT_ANSI;
148              else if(!strcmp(word, "iso"))
149                  kb->features = (kb->features & ~FEAT_LMASK) |
      FEAT_ISO;
150              continue;
151  #ifdef OS_MAC
152          case ACCEL:
153              // OSX mouse acceleration on/off
154              if(!strcmp(word, "on"))
155                  kb->features |= FEAT_MOUSEACCEL;
156              else if(!strcmp(word, "off"))
157                  kb->features &= ~FEAT_MOUSEACCEL;
158              continue;
159          case SCROLLSPEED:{
160              int newscroll;
161              if(sscanf(word, "%d", &newscroll) != 1)
162                  break;
163              if(newscroll < SCROLL_MIN)
```

```
164                 newscroll = SCROLL_ACCELERATED;
165             if(newscroll > SCROLL_MAX)
166                 newscroll = SCROLL_MAX;
167             kb->scroll_rate = newscroll;
168             continue;
169         }
170 #endif
171         case MODE: {
172             // Select a mode number (1 - 6)
173             int newmode;
174             if(sscanf(word, "%u", &newmode) == 1 && newmode > 0 && newmode <=
    MODE_COUNT)
175                 mode = profile->mode + newmode - 1;
176             continue;
177         }
178         case FPS: {
179             // USB command delay (2 - 10ms)
180             uint framerate;
181             if(sscanf(word, "%u", &framerate) == 1 && framerate > 0){
182                 // Not all devices require the same number of messages per frame; select delay
    appropriately
183                 uint per_frame = IS_MOUSE_DEV(kb) ? 2 : IS_FULLRANGE(kb) ? 14 : 5;
184                 uint delay = 1000 / framerate / per_frame;
185                 if(delay < 2)
186                     delay = 2;
187                 else if(delay > 10)
188                     delay = 10;
189                 kb->usbdelay = delay;
190             }
191             continue;
192         }
193         case DITHER: {
194             // 0: No dither, 1: Ordered dither.
195             uint dither;
196             if(sscanf(word, "%u", &dither) == 1 && dither <= 1){
197                 kb->dither = dither;
198                 profile->currentmode->light.forceupdate = 1;
199                 mode->light.forceupdate = 1;
200             }
201             continue;
202         }
203         case DELAY:
204             kb->delay = (!strcmp (word, "on")); // independendant from parameter to handle false
    commands like "delay off"
205             continue;
206         case RESTART: {
207             char mybuffer[] = "no reason specified";
208             if (sscanf(line, " %[^\n]", word) == -1) {
209                 word = mybuffer;
210             }
211             vt->do_cmd[command](kb, mode, notifynumber, 0, word);
212             continue;
213         }
214
215         default:;
216         }
217
218         // If a keyboard is inactive, it must be activated before receiving any other commands
219         if(!kb->active){
220             if(command == ACTIVE)
221                 TRY_WITH_RESET(vt->active(kb, mode, notifynumber, 0, 0));
222             continue;
223         }
224         // Specially handled commands only available when keyboard is ACTIVE
225         switch(command){
226         case IDLE:
227             TRY_WITH_RESET(vt->idle(kb, mode, notifynumber, 0, 0));
228             continue;
229         case SWITCH:
230             if(profile->currentmode != mode){
231                 profile->currentmode = mode;
232                 // Set mode light for non-RGB K95
233                 int index = INDEX_OF(mode, profile->mode);
234                 vt->setmodeindex(kb, index);
235             }
236             continue;
237         case HWLOAD: case HWSAVE:{
238             char delay = kb->usbdelay;
239             // Ensure delay of at least 10ms as the device can get overwhelmed otherwise
240             if(delay < 10)
241                 kb->usbdelay = 10;
242             // Try to load/save the hardware profile. Reset on failure, disconnect if reset fails.
243             TRY_WITH_RESET(vt->do_io[command](kb, mode, notifynumber, 1, 0));
244             // Re-send the current RGB state as it sometimes gets scrambled
245             TRY_WITH_RESET(vt->updatergb(kb, 1));
246             kb->usbdelay = delay;
247             continue;
```

```
248         }
249         case FWUPDATE:
250             // FW update parses a whole word. Unlike hwload/hwsave, there's no try again on failure.
251             if(vt->fwupdate(kb, mode, notifynumber, 0, word)){
252                 free(word);
253                 return 1;
254             }
255             continue;
256         case POLLRATE: {
257             uint rate;
258             if(sscanf(word, "%u", &rate) == 1 && (rate == 1 || rate == 2 || rate == 4 || rate == 8))
259                 TRY_WITH_RESET(vt->pollrate(kb, mode, notifynumber, rate, 0));
260             continue;
261         }
262         case ERASEPROFILE:
263             // Erase the current profile
264             vt->eraseprofile(kb, mode, notifynumber, 0, 0);
265             // Update profile/mode pointers
266             profile = kb->profile;
267             mode = profile->currentmode;
268             continue;
269         case ERASE: case NAME: case IOFF: case ION: case IAUTO: case
    INOTIFY: case PROFILENAME: case ID: case PROFILEID: case
    DPISEL: case LIFT: case SNAP:
270             // All of the above just parse the whole word
271             vt->do_cmd[command](kb, mode, notifynumber, 0, word);
272             continue;
273         case RGB: {
274             // RGB command has a special response for a single hex constant
275             int r, g, b;
276             if(sscanf(word, "%02x%02x%02x", &r, &g, &b) == 3){
277                 // Set all keys
278                 for(int i = 0; i < N_KEYS_EXTENDED; i++)
279                     vt->rgb(kb, mode, notifynumber, i, word);
280                 continue;
281             }
282             break;
283         }
284         case MACRO:
285             if(!strcmp(word, "clear")){
286                 // Macro has a special clear command
287                 vt->macro(kb, mode, notifynumber, 0, 0);
288                 continue;
289             }
290             break;
291         default:;
292         }
293         // For anything else, split the parameter at the colon
294         int left = -1;
295         sscanf(word, "%*[^:]%n", &left);
296         if(left <= 0)
297             continue;
298         const char* right = word + left;
299         if(right[0] == ':')
300             right++;
301         // Macros and DPI have a separate left-side handler
302         if(command == MACRO || command == DPI){
303             word[left] = 0;
304             vt->do_macro[command](kb, mode, notifynumber, word, right);
305             continue;
306         }
307         // Scan the left side for key names and run the requested command
308         int position = 0, field = 0;
309         char keyname[11];
310         while(position < left && sscanf(word + position, "%10[^:,]%n", keyname, &field) == 1){
311             int keycode;
312             if(!strcmp(keyname, "all")){
313                 // Set all keys
314                 for(int i = 0; i < N_KEYS_EXTENDED; i++)
315                     vt->do_cmd[command](kb, mode, notifynumber, i, right);
316             } else if((sscanf(keyname, "#%d", &keycode) && keycode >= 0 && keycode <
    N_KEYS_EXTENDED)
317                       || (sscanf(keyname, "#x%x", &keycode) && keycode >= 0 && keycode <
    N_KEYS_EXTENDED)){
318                 // Set a key numerically
319                 vt->do_cmd[command](kb, mode, notifynumber, keycode, right);
320             } else {
321                 // Find this key in the keymap
322                 for(unsigned i = 0; i < N_KEYS_EXTENDED; i++){
323                     if(keymap[i].name && !strcmp(keyname, keymap[i].name)){
324                         vt->do_cmd[command](kb, mode, notifynumber, i, right);
325                         break;
326                     }
327                 }
328             }
329             if(word[position += field] == ',')
330                 position++;
```

```
331         }
332     }
333
334     // Finish up
335     if(!NEEDS_FW_UPDATE(kb)){
336         TRY_WITH_RESET(vt->updatergb(kb, 0));
337         TRY_WITH_RESET(vt->updatedpi(kb, 0));
338     }
339     free(word);
340     return 0;
341 }
```

Here is the call graph for this function:



Here is the caller graph for this function:

8.5.3 Variable Documentation

8.5.3.1 const char* const cmd_strings[(CMD_LAST-CMD_FIRST+2)-1] [static]

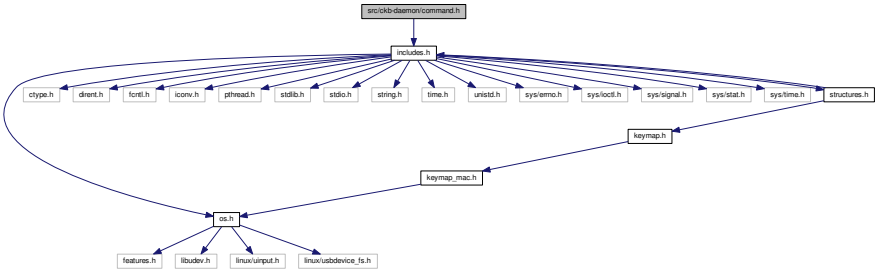
Definition at line 9 of file command.c.

Referenced by readcmd().

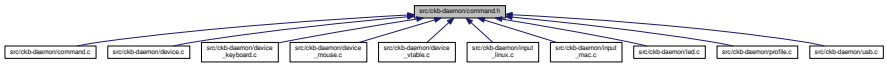
8.6 src/ckb-daemon/command.h File Reference

```
#include "includes.h"
```

Include dependency graph for command.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- union devcmd
- struct devcmd.__unnamed__

Macros

- #define CMD_COUNT (CMD_LAST - CMD_FIRST + 2)
- #define CMD_DEV_COUNT (CMD_LAST - CMD_VT_FIRST + 1)

Typedefs

- typedef void(∗ cmdhandler)(usbdevice ∗kb, usbmode ∗modeidx, int notifyidx, int keyindex, const char ∗parameter)
- typedef int(∗ cmdhandler_io)(usbdevice ∗kb, usbmode ∗modeidx, int notifyidx, int keyindex, const char ∗parameter)
- typedef void(∗ cmdhandler_mac)(usbdevice ∗kb, usbmode ∗modeidx, int notifyidx, const char ∗keys, const char ∗assignment)
- typedef union devcmd devcmd

Enumerations

- enum cmd {
  NONE = -11, DELAY = -10, CMD_FIRST = DELAY, MODE = -9,
  SWITCH = -8, LAYOUT = -7, ACCEL = -6, SCROLLSPEED = -5,
  NOTIFYON = -4, NOTIFYOFF = -3, FPS = -2, DITHER = -1,
  HWLOAD = 0, CMD_VT_FIRST = 0, HWSAVE, FWUPDATE,
  POLLRATE, ACTIVE, IDLE, ERASE,
  ERASEPROFILE, NAME, PROFILENAME, ID,
  PROFILEID, RGB, IOFF, ION,
  IAUTO, BIND, UNBIND, REBIND,
  MACRO, DPI, DPISEL, LIFT,
  SNAP, NOTIFY, INOTIFY, GET,
  RESTART, CMD_LAST = RESTART }

Functions

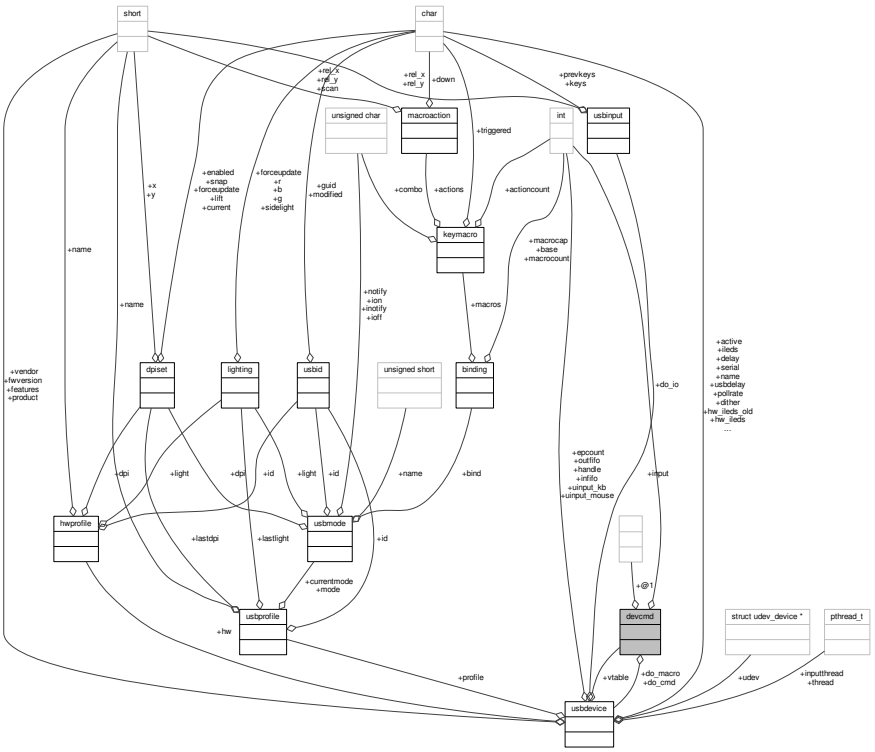
- int readcmd (usbdevice ∗kb, const char ∗line)

8.6.1 Data Structure Documentation

8.6.1.1 union devcmd

Definition at line 73 of file command.h.

Collaboration diagram for devcmd:



Data Fields

struct devcmd	__unnamed__	
cmdhandler	do_cmd[(CMD_-LAST-CMD_VT-_FIRST+1)]	
cmdhandler_io	do_io[(CMD_LA-ST-CMD_VT_FI-RST+1)]	
cmdhandler_-mac	do_macro[(CM-D_LAST-CMD_-VT_FIRST+1)]	

8.6.2 Macro Definition Documentation

8.6.2.1 #define CMD_COUNT (CMD_LAST - CMD_FIRST + 2)

Definition at line 65 of file command.h.

Referenced by readcmd().

8.6.2.2 #define CMD_DEV_COUNT (CMD_LAST - CMD_VT_FIRST + 1)

Definition at line 66 of file command.h.

8.6.3 Typedef Documentation

8.6.3.1 typedef void(∗ cmdhandler)(usbdevice ∗kb, usbmode ∗modeidx, int notifyidx, int keyindex, const char ∗parameter)

Definition at line 70 of file command.h.

8.6.3.2 typedef int(∗ cmdhandler_io)(usbdevice ∗kb, usbmode ∗modeidx, int notifyidx, int keyindex, const char ∗parameter)

Definition at line 71 of file command.h.

8.6.3.3 typedef void(∗ cmdhandler_mac)(usbdevice ∗kb, usbmode ∗modeidx, int notifyidx, const char ∗keys, const char ∗assignment)

Definition at line 72 of file command.h.

8.6.3.4 typedef union devcmd devcmd

8.6.4 Enumeration Type Documentation

8.6.4.1 enum cmd

Enumerator

NONE

DELAY

CMD_FIRST

MODE

SWITCH

LAYOUT

ACCEL

SCROLLSPEED

NOTIFYON

NOTIFYOFF

FPS

DITHER

HWLOAD

CMD_VT_FIRST

HWSAVE

FWUPDATE

POLLRATE

ACTIVE

IDLE

ERASE

ERASEPROFILE

NAME

PROFILENAME

ID

PROFILEID

RGB

IOFF

ION

IAUTO

BIND

UNBIND

REBIND

MACRO

DPI

DPISEL

LIFT

SNAP

NOTIFY

INOTIFY

GET

RESTART

CMD_LAST

Definition at line 7 of file command.h.

```
7             {
8     // Special - handled by readcmd, no device functions
9     NONE        = -11,
10     DELAY       = -10,   CMD_FIRST = DELAY,
11     MODE        = -9,
12     SWITCH      = -8,
13     LAYOUT      = -7,
14     ACCEL       = -6,
15     SCROLLSPEED = -5,
16     NOTIFYON    = -4,
17     NOTIFYOFF   = -3,
18     FPS         = -2,
19     DITHER      = -1,
20
21     // Hardware data
22     HWLOAD      = 0,    CMD_VT_FIRST = 0,
23     HWSAVE,
24     FWUPDATE,
25     POLLRATE,
26
27     // Software control on/off
28     ACTIVE,
29     IDLE,
30
31     // Profile/mode metadata
32     ERASE,
33     ERASEPROFILE,
34     NAME,
35     PROFILENAME,
36     ID,
37     PROFILEID,
38
39     // LED control
40     RGB,
41     IOFF,
42     ION,
43     IAUTO,
44
45     // Key binding control
46     BIND,
47     UNBIND,
48     REBIND,
49     MACRO,
50
51     // DPI control
52     DPI,
53     DPISEL,
54     LIFT,
55     SNAP,
56
57     // Notifications and output
58     NOTIFY,
59     INOTIFY,
```

```
60     GET,
61     RESTART,
62
63     CMD_LAST = RESTART
64 } cmd;
```

8.6.5 Function Documentation

8.6.5.1 int readcmd (usbdevice * kb, const char * line)

Because length of word is length of line + 1, there should be no problem with buffer overflow.

Definition at line 67 of file command.c.

References ACCEL, ACTIVE, usbdevice::active, BIND, CMD_COUNT, CMD_FIRST, cmd_strings, usbprofile::currentmode, DELAY, usbdevice::delay, DITHER, usbdevice::dither, devcmd::do_cmd, devcmd::do_io, devcmd::do_macro, DPI, DPISEL, ERASE, ERASEPROFILE, FEAT_ANSI, FEAT_BIND, FEAT_ISO, FEAT_LMASK, FEAT_MOUSEACCEL, FEAT_NOTIFY, usbdevice::features, lighting::forceupdate, FPS, FWUPDATE, GET, HAS_FEATURES, HWLOAD, HWSAVE, IAUTO, ID, IDLE, INDEX_OF, INOTIFY, IOFF, ION, IS_FULLRANGE, IS_MOUSE_DEV, keymap, LAYOUT, LIFT, usbmode::light, MACRO, mknotifynode(), MODE, usbprofile::mode, MODE_COUNT, N_KEYS_EXTENDED, NAME, NEEDS_FW_UPDATE, NONE, NOTIFY, NOTIFYOFF, NOTIFYON, OUTFIFO_MAX, POLLRATE, usbdevice::profile, PROFILEID, PROFILENAME, REBIND, RESTART, RGB, rmnotifynode(), SCROLL_ACCELERATED, SCROLL_MAX, SCROLL_MIN, SCROLLSPEED, SNAP, SWITCH, TRY_WITH_RESET, UNBIND, usbdevice::usbdelay, and usbdevice::vtable.

Referenced by devmain().

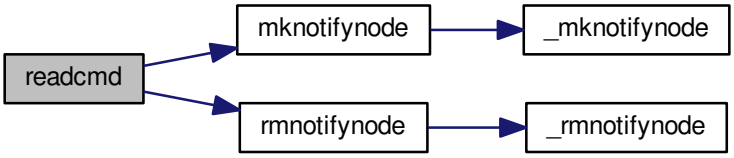
```
67                                          {
68     char* word = malloc(strlen(line) + 1);
69     int wordlen;
70     const char* newline = 0;
71     const devcmd* vt = kb->vtable;
72     usbprofile* profile = kb->profile;
73     usbmode* mode = 0;
74     int notifynumber = 0;
75     // Read words from the input
76     cmd command = NONE;
77     while(sscanf(line, "%s%n", word, &wordlen) == 1){
78         line += wordlen;
79         // If we passed a newline, reset the context
80         if(line > newline){
81             mode = profile->currentmode;
82             command = NONE;
83             notifynumber = 0;
84             newline = strchr(line, '\n');
85             if(!newline)
86                 newline = line + strlen(line);
87         }
88         // Check for a command word
89         for(int i = 0; i < CMD_COUNT - 1; i++){
90             if(!strcmp(word, cmd_strings[i])){
91                 command = i + CMD_FIRST;
92 #ifndef OS_MAC
93                 // Layout and mouse acceleration aren't used on Linux; ignore
94                 if(command == LAYOUT || command == ACCEL || command ==
    SCROLLSPEED)
95                     command = NONE;
96 #endif
97                 // Most commands require parameters, but a few are actions in and of themselves
98                 if(command != SWITCH
99                         && command != HWLOAD && command != HWSAVE
100                         && command != ACTIVE && command != IDLE
101                         && command != ERASE && command != ERASEPROFILE
102                         && command != RESTART)
103                     goto next_loop;
104                 break;
105             }
106         }
107
108         // Set current notification node when given @number
109         int newnotify;
110         if(sscanf(word, "@%u", &newnotify) == 1 && newnotify < OUTFIFO_MAX){
111             notifynumber = newnotify;
112             continue;
113         }
114
```

```
115         // Reject unrecognized commands. Reject bind or notify related commands if the keyboard doesn't
        have the feature enabled.
116         if(command == NONE
117                 || ((!HAS_FEATURES(kb, FEAT_BIND) && (command ==
        BIND || command == UNBIND || command == REBIND || command ==
        MACRO || command == DELAY))
118                         || (!HAS_FEATURES(kb, FEAT_NOTIFY) && command ==
        NOTIFY))){
119             next_loop:
120             continue;
121         }
122         // Reject anything not related to fwupdate if device has a bricked FW
123         if(NEEDS_FW_UPDATE(kb) && command != FWUPDATE && command !=
        NOTIFYON && command != NOTIFYOFF)
124             continue;
125
126         // Specially handled commands - these are available even when keyboard is IDLE
127         switch(command){
128         case NOTIFYON: {
129             // Notification node on
130             int notify;
131             if(sscanf(word, "%u", &notify) == 1)
132                 mknotifynode(kb, notify);
133             continue;
134         } case NOTIFYOFF: {
135             // Notification node off
136             int notify;
137             if(sscanf(word, "%u", &notify) == 1 && notify != 0) // notify0 can't be removed
138                 rmnotifynode(kb, notify);
139             continue;
140         } case GET:
141             // Output data to notification node
142             vt->get(kb, mode, notifynumber, 0, word);
143             continue;
144         case LAYOUT:
145             // OSX: switch ANSI/ISO keyboard layout
146             if(!strcmp(word, "ansi"))
147                 kb->features = (kb->features & ~FEAT_LMASK) |
        FEAT_ANSI;
148             else if(!strcmp(word, "iso"))
149                 kb->features = (kb->features & ~FEAT_LMASK) |
        FEAT_ISO;
150             continue;
151 #ifdef OS_MAC
152         case ACCEL:
153             // OSX mouse acceleration on/off
154             if(!strcmp(word, "on"))
155                 kb->features |= FEAT_MOUSEACCEL;
156             else if(!strcmp(word, "off"))
157                 kb->features &= ~FEAT_MOUSEACCEL;
158             continue;
159         case SCROLLSPEED:{
160             int newscroll;
161             if(sscanf(word, "%d", &newscroll) != 1)
162                 break;
163             if(newscroll < SCROLL_MIN)
164                 newscroll = SCROLL_ACCELERATED;
165             if(newscroll > SCROLL_MAX)
166                 newscroll = SCROLL_MAX;
167             kb->scroll_rate = newscroll;
168             continue;
169         }
170 #endif
171         case MODE: {
172             // Select a mode number (1 - 6)
173             int newmode;
174             if(sscanf(word, "%u", &newmode) == 1 && newmode > 0 && newmode <=
        MODE_COUNT)
175                 mode = profile->mode + newmode - 1;
176             continue;
177         }
178         case FPS: {
179             // USB command delay (2 - 10ms)
180             uint framerate;
181             if(sscanf(word, "%u", &framerate) == 1 && framerate > 0){
182                 // Not all devices require the same number of messages per frame; select delay
        appropriately
183                 uint per_frame = IS_MOUSE_DEV(kb) ? 2 : IS_FULLRANGE(kb) ? 14 : 5;
184                 uint delay = 1000 / framerate / per_frame;
185                 if(delay < 2)
186                     delay = 2;
187                 else if(delay > 10)
188                     delay = 10;
189                 kb->usbdelay = delay;
190             }
191             continue;
192         }
```

```
193         case DITHER: {
194             // 0: No dither, 1: Ordered dither.
195             uint dither;
196             if(sscanf(word, "%u", &dither) == 1 && dither <= 1){
197                 kb->dither = dither;
198                 profile->currentmode->light.forceupdate = 1;
199                 mode->light.forceupdate = 1;
200             }
201             continue;
202         }
203         case DELAY:
204             kb->delay = (!strcmp (word, "on")); // independendant from parameter to handle false
    commands like "delay off"
205             continue;
206         case RESTART: {
207             char mybuffer[] = "no reason specified";
208             if (sscanf(line, " %[^\n]", word) == -1) {
209                 word = mybuffer;
210             }
211             vt->do_cmd[command](kb, mode, notifynumber, 0, word);
212             continue;
213         }
214 
215         default:;
216         }
217 
218         // If a keyboard is inactive, it must be activated before receiving any other commands
219         if(!kb->active){
220             if(command == ACTIVE)
221                 TRY_WITH_RESET(vt->active(kb, mode, notifynumber, 0, 0));
222             continue;
223         }
224         // Specially handled commands only available when keyboard is ACTIVE
225         switch(command){
226         case IDLE:
227             TRY_WITH_RESET(vt->idle(kb, mode, notifynumber, 0, 0));
228             continue;
229         case SWITCH:
230             if(profile->currentmode != mode){
231                 profile->currentmode = mode;
232                 // Set mode light for non-RGB K95
233                 int index = INDEX_OF(mode, profile->mode);
234                 vt->setmodeindex(kb, index);
235             }
236             continue;
237         case HWLOAD: case HWSAVE:{
238             char delay = kb->usbdelay;
239             // Ensure delay of at least 10ms as the device can get overwhelmed otherwise
240             if(delay < 10)
241                 kb->usbdelay = 10;
242             // Try to load/save the hardware profile. Reset on failure, disconnect if reset fails.
243             TRY_WITH_RESET(vt->do_io[command](kb, mode, notifynumber, 1, 0));
244             // Re-send the current RGB state as it sometimes gets scrambled
245             TRY_WITH_RESET(vt->updatergb(kb, 1));
246             kb->usbdelay = delay;
247             continue;
248         }
249         case FWUPDATE:
250             // FW update parses a whole word. Unlike hwload/hwsave, there's no try again on failure.
251             if(vt->fwupdate(kb, mode, notifynumber, 0, word)){
252                 free(word);
253                 return 1;
254             }
255             continue;
256         case POLLRATE: {
257             uint rate;
258             if(sscanf(word, "%u", &rate) == 1 && (rate == 1 || rate == 2 || rate == 4 || rate == 8))
259                 TRY_WITH_RESET(vt->pollrate(kb, mode, notifynumber, rate, 0));
260             continue;
261         }
262         case ERASEPROFILE:
263             // Erase the current profile
264             vt->eraseprofile(kb, mode, notifynumber, 0, 0);
265             // Update profile/mode pointers
266             profile = kb->profile;
267             mode = profile->currentmode;
268             continue;
269         case ERASE: case NAME: case IOFF: case ION: case IAUTO: case
    INOTIFY: case PROFILENAME: case ID: case PROFILEID: case
    DPISEL: case LIFT: case SNAP:
270             // All of the above just parse the whole word
271             vt->do_cmd[command](kb, mode, notifynumber, 0, word);
272             continue;
273         case RGB: {
274             // RGB command has a special response for a single hex constant
275             int r, g, b;
276             if(sscanf(word, "%02x%02x%02x", &r, &g, &b) == 3){
```

```
277                 // Set all keys
278                 for(int i = 0; i < N_KEYS_EXTENDED; i++)
279                     vt->rgb(kb, mode, notifynumber, i, word);
280                 continue;
281             }
282             break;
283         }
284         case MACRO:
285             if(!strcmp(word, "clear")){
286                 // Macro has a special clear command
287                 vt->macro(kb, mode, notifynumber, 0, 0);
288                 continue;
289             }
290             break;
291         default:;
292         }
293         // For anything else, split the parameter at the colon
294         int left = -1;
295         sscanf(word, "%*[^:]%n", &left);
296         if(left <= 0)
297             continue;
298         const char* right = word + left;
299         if(right[0] == ':')
300             right++;
301         // Macros and DPI have a separate left-side handler
302         if(command == MACRO || command == DPI){
303             word[left] = 0;
304             vt->do_macro[command](kb, mode, notifynumber, word, right);
305             continue;
306         }
307         // Scan the left side for key names and run the requested command
308         int position = 0, field = 0;
309         char keyname[11];
310         while(position < left && sscanf(word + position, "%10[^:,]%n", keyname, &field) == 1){
311             int keycode;
312             if(!strcmp(keyname, "all")){
313                 // Set all keys
314                 for(int i = 0; i < N_KEYS_EXTENDED; i++)
315                     vt->do_cmd[command](kb, mode, notifynumber, i, right);
316             } else if((sscanf(keyname, "#%d", &keycode) && keycode >= 0 && keycode <
    N_KEYS_EXTENDED)
317                       || (sscanf(keyname, "#x%x", &keycode) && keycode >= 0 && keycode <
    N_KEYS_EXTENDED)){
318                 // Set a key numerically
319                 vt->do_cmd[command](kb, mode, notifynumber, keycode, right);
320             } else {
321                 // Find this key in the keymap
322                 for(unsigned i = 0; i < N_KEYS_EXTENDED; i++){
323                     if(keymap[i].name && !strcmp(keyname, keymap[i].name)){
324                         vt->do_cmd[command](kb, mode, notifynumber, i, right);
325                         break;
326                     }
327                 }
328             }
329             if(word[position += field] == ',')
330                 position++;
331         }
332     }
333
334     // Finish up
335     if(!NEEDS_FW_UPDATE(kb)){
336         TRY_WITH_RESET(vt->updatergb(kb, 0));
337         TRY_WITH_RESET(vt->updatedpi(kb, 0));
338     }
339     free(word);
340     return 0;
341 }
```

Here is the call graph for this function:

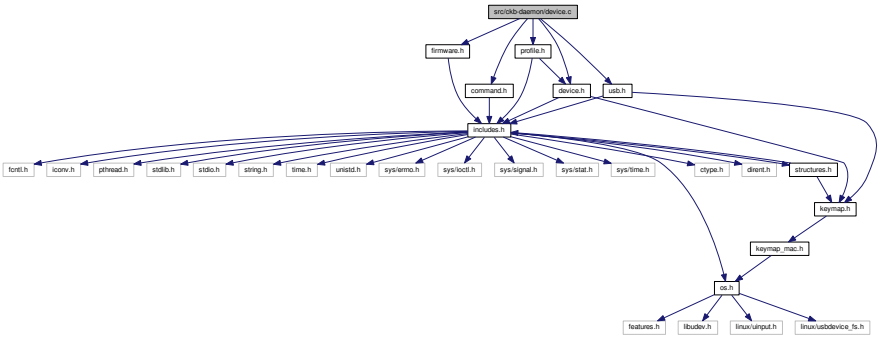


Here is the caller graph for this function:

8.7 src/ckb-daemon/device.c File Reference

```
#include "command.h"
#include "device.h"
#include "firmware.h"
#include "profile.h"
#include "usb.h"
```

Include dependency graph for device.c:



Functions

- int _start_dev (usbdevice *kb, int makeactive)
- int start_dev (usbdevice *kb, int makeactive)

Variables

- int hwload_mode = 1
- usbdevice keyboard [9]

- pthread_mutex_t devlistmutex = PTHREAD_MUTEX_INITIALIZER
- pthread_mutex_t devmutex [9] = { [0 ... 9 -1] = PTHREAD_MUTEX_INITIALIZER }
- pthread_mutex_t inputmutex [9] = { [0 ... 9 -1] = PTHREAD_MUTEX_INITIALIZER }

8.7.1 Function Documentation

8.7.1.1 int _start_dev (usbdevice * *kb*, int *makeactive*)

Definition at line 15 of file device.c.

References usbdevice::active, ckb_info, ckb_warn, FEAT_ADJRATE, FEAT_FWUPDATE, FEAT_FWVERSION, FEAT_HWLOAD, FEAT_POLLRATE, FEAT_RGB, usbdevice::features, usbdevice::fwversion, getfwversion(), HAS_FEATURES, usbdevice::hw, hwload_mode, hwloadprofile, NEEDS_FW_UPDATE, usbdevice::pollrate, and setactive.

Referenced by start_dev().

```
15                                          {
16     // Get the firmware version from the device
17     if(kb->pollrate == 0){
18         if(!hwload_mode || (HAS_FEATURES(kb, FEAT_HWLOAD) &&
    getfwversion(kb))){
19             if(hwload_mode == 2)
20                 // hwload=always. Report setup failure.
21                 return -1;
22             else if(hwload_mode){
23                 // hwload=once. Log failure, prevent trying again, and continue.
24                 ckb_warn("Unable to load firmware version/poll rate\n");
25                 kb->features &= ~FEAT_HWLOAD;
26             }
27             kb->pollrate = 0;
28             kb->features &= ~(FEAT_POLLRATE | FEAT_ADJRATE);
29             if(kb->fwversion == 0)
30                 kb->features &= ~(FEAT_FWVERSION |
    FEAT_FWUPDATE);
31         }
32     }
33     if(NEEDS_FW_UPDATE(kb)){
34         // Device needs a firmware update. Finish setting up but don't do anything.
35         ckb_info("Device needs a firmware update. Please issue a fwupdate command.\n");
36         kb->features = FEAT_RGB | FEAT_FWVERSION |
    FEAT_FWUPDATE;
37         kb->active = 1;
38         return 0;
39     }
40     // Load profile from device
41     if(!kb->hw && hwload_mode && HAS_FEATURES(kb,
    FEAT_HWLOAD)){
42         if(hwloadprofile(kb, 1)){
43             if(hwload_mode == 2)
44                 return -1;
45             ckb_warn("Unable to load hardware profile\n");
46             kb->features &= ~FEAT_HWLOAD;
47         }
48     }
49     // Active software mode if requested
50     if(makeactive)
51         return setactive(kb, 1);
52     return 0;
53 }
```

Here is the call graph for this function:

Here is the caller graph for this function:

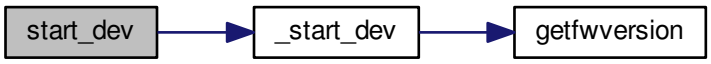
8.7.1.2 int start_dev (usbdevice * kb, int makeactive)

Definition at line 55 of file device.c.

References _start_dev(), USB_DELAY_DEFAULT, and usbdevice::usbdelay.

```
55                                          {
56     // Force USB interval to 10ms during initial setup phase; return to nominal 5ms after setup completes.
57     kb->usbdelay = 10;
58     int res = _start_dev(kb, makeactive);
59     kb->usbdelay = USB_DELAY_DEFAULT;
60     return res;
61 }
```

Here is the call graph for this function:



8.7.2 Variable Documentation

8.7.2.1 pthread_mutex_t devlistmutex = PTHREAD_MUTEX_INITIALIZER

Definition at line 11 of file device.c.

8.7.2.2 pthread_mutex_t devmutex[9] = { [0 ... 9 -1] = PTHREAD_MUTEX_INITIALIZER }

Definition at line 12 of file device.c.

Referenced by _updateconnected(), quitWithLock(), and usb_rm_device().

8.7.2.3 int hwload_mode = 1

Definition at line 7 of file device.c.

Referenced by _start_dev(), _usbrecv(), _usbsend(), and main().

8.7.2.4 pthread_mutex_t inputmutex[9] = { [0 ... 9 -1] = PTHREAD_MUTEX_INITIALIZER }

Definition at line 13 of file device.c.

8.7.2.5 usbdevice keyboard[9]

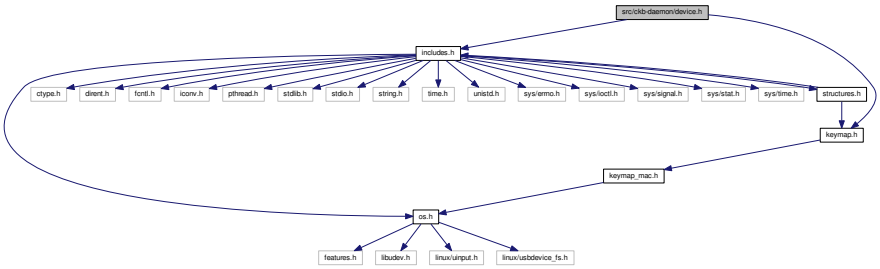
Definition at line 10 of file device.c.

Referenced by _mkdevpath(), _mknotifynode(), _rmnotifynode(), _setupusb(), _updateconnected(), closeusb(), main(), mkfwnode(), os_closeusb(), os_inputmain(), os_inputopen(), os_setupusb(), quitWithLock(), rmdevpath(), usb_rm_device(), and usbadd().

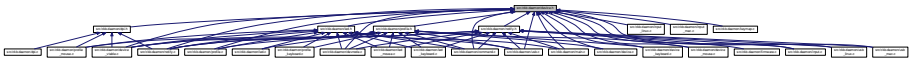
8.8 src/ckb-daemon/device.h File Reference

```
#include "includes.h"
#include "keymap.h"
```

Include dependency graph for device.h:



This graph shows which files directly or indirectly include this file:



Macros

- #define DEV_MAX 9
- #define IS_CONNECTED(kb) ((kb) && (kb)->handle && (kb)->uinput_kb && (kb)->uinput_mouse)
- #define dmutex(kb) (devmutex + INDEX_OF(kb, keyboard))
- #define imutex(kb) (inputmutex + INDEX_OF(kb, keyboard))
- #define setactive(kb, makeactive) ((makeactive) ? (kb)->vtable->active((kb), 0, 0, 0, 0) : (kb)->vtable->idle((kb), 0, 0, 0, 0))
- #define IN_HID 0x80
- #define IN_CORSAIR 0x40
- #define ACT_LIGHT 1
- #define ACT_NEXT 3
- #define ACT_NEXT_NOWRAP 5
- #define ACT_LOCK 8
- #define ACT_MR_RING 9
- #define ACT_M1 10
- #define ACT_M2 11
- #define ACT_M3 12

Functions

- int start_dev (usbdevice *kb, int makeactive)
- int start_kb_nrgb (usbdevice *kb, int makeactive)
- int setactive_kb (usbdevice *kb, int active)
- int setactive_mouse (usbdevice *kb, int active)
- int cmd_active_kb (usbdevice *kb, usbmode *dummy1, int dummy2, int dummy3, const char *dummy4)
- int cmd_active_mouse (usbdevice *kb, usbmode *dummy1, int dummy2, int dummy3, const char *dummy4)
- int cmd_idle_kb (usbdevice *kb, usbmode *dummy1, int dummy2, int dummy3, const char *dummy4)
- int cmd_idle_mouse (usbdevice *kb, usbmode *dummy1, int dummy2, int dummy3, const char *dummy4)
- int cmd_pollrate (usbdevice *kb, usbmode *dummy1, int dummy2, int rate, const char *dummy3)
- void setmodeindex_nrgb (usbdevice *kb, int index)

Variables

- usbdevice keyboard [9]
- pthread_mutex_t devmutex [9]
- pthread_mutex_t inputmutex [9]

8.8.1 Macro Definition Documentation

8.8.1.1 #define ACT_LIGHT 1

Definition at line 56 of file device.h.

Referenced by setactive_kb().

8.8.1.2 #define ACT_LOCK 8

Definition at line 59 of file device.h.

Referenced by setactive_kb().

8.8.1.3 #define ACT_M1 10

Definition at line 61 of file device.h.

Referenced by setactive_kb().

8.8.1.4 #define ACT_M2 11

Definition at line 62 of file device.h.

Referenced by setactive_kb().

8.8.1.5 #define ACT_M3 12

Definition at line 63 of file device.h.

Referenced by setactive_kb().

8.8.1.6 #define ACT_MR_RING 9

Definition at line 60 of file device.h.

Referenced by setactive_kb().

8.8.1.7 #define ACT_NEXT 3

Definition at line 57 of file device.h.

8.8.1.8 #define ACT_NEXT_NOWRAP 5

Definition at line 58 of file device.h.

8.8.1.9 #define DEV_MAX 9

Definition at line 8 of file device.h.

Referenced by _updateconnected(), quitWithLock(), usb_rm_device(), and usbadd().

8.8.1.10 #define dmutex(*kb*) (devmutex + INDEX_OF(kb, keyboard))

Definition at line 18 of file device.h.

Referenced by _ledthread(), _setupusb(), closeusb(), devmain(), and usbadd().

8.8.1.11 #define imutex(*kb*) (inputmutex + INDEX_OF(kb, keyboard))

Definition at line 22 of file device.h.

Referenced by _setupusb(), closeusb(), cmd_bind(), cmd_erase(), cmd_eraseprofile(), cmd_get(), cmd_macro(), cmd_notify(), cmd_rebind(), cmd_unbind(), os_inputmain(), setactive_kb(), setactive_mouse(), and setupusb().

8.8.1.12 #define IN_CORSAIR 0x40

Definition at line 53 of file device.h.

Referenced by setactive_kb(), and setactive_mouse().

8.8.1.13 #define IN_HID 0x80

Definition at line 52 of file device.h.

Referenced by setactive_kb(), and setactive_mouse().

8.8.1.14 #define IS_CONNECTED(*kb*) ((kb) && (kb)->handle && (kb)->uinput_kb && (kb)->uinput_mouse)

Definition at line 12 of file device.h.

Referenced by _updateconnected(), devmain(), quitWithLock(), and usbadd().

8.8.1.15 #define setactive(*kb, makeactive*) ((makeactive) ? (kb)->vtable->active((kb), 0, 0, 0, 0) : (kb)->vtable->idle((kb), 0, 0, 0, 0))

Definition at line 32 of file device.h.

Referenced by _start_dev(), and revertusb().

8.8.2 Function Documentation

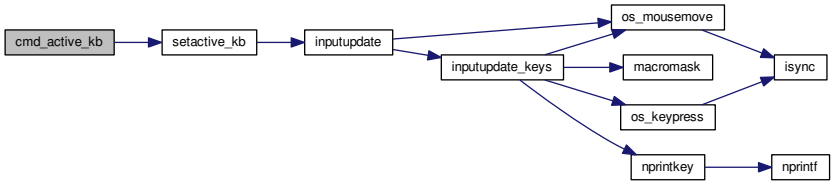
8.8.2.1 int cmd_active_kb (usbdevice * kb, usbmode * dummy1, int dummy2, int dummy3, const char * dummy4)

Definition at line 112 of file device_keyboard.c.

References setactive_kb().

```
112                                                                                 {
113     return setactive_kb(kb, 1);
114 }
```

Here is the call graph for this function:



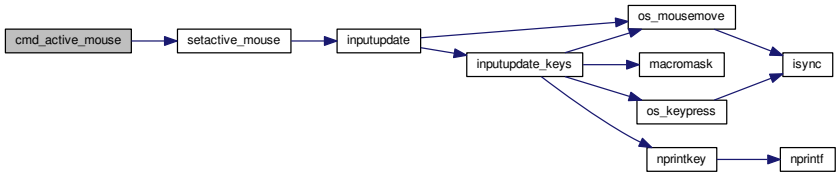
8.8.2.2 int cmd_active_mouse (usbdevice * kb, usbmode * dummy1, int dummy2, int dummy3, const char * dummy4)

Definition at line 44 of file device_mouse.c.

References setactive_mouse().

```
44                                                                                  {
45     return setactive_mouse(kb, 1);
46 }
```

Here is the call graph for this function:



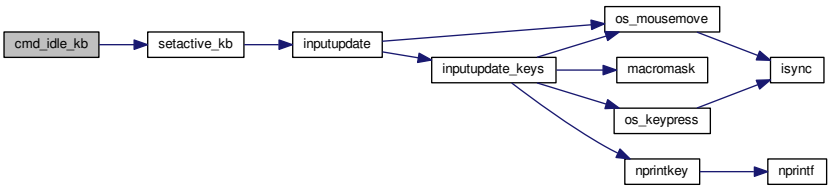
8.8.2.3 int cmd_idle_kb (usbdevice * kb, usbmode * dummy1, int dummy2, int dummy3, const char * dummy4)

Definition at line 116 of file device_keyboard.c.

References setactive_kb().

```
116                                                                                 {
117     return setactive_kb(kb, 0);
118 }
```

Here is the call graph for this function:



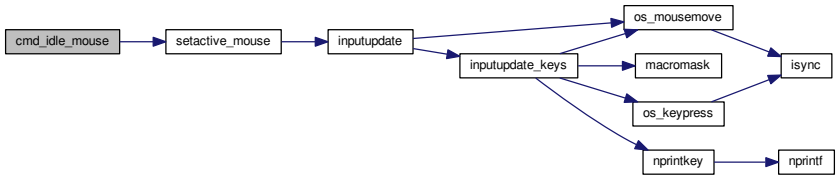
8.8.2.4 int cmd_idle_mouse (usbdevice * *kb*, usbmode * *dummy1*, int *dummy2*, int *dummy3*, const char * *dummy4*)

Definition at line 48 of file device_mouse.c.

References setactive_mouse().

```
48                                                                                  {
49     return setactive_mouse(kb, 0);
50 }
```

Here is the call graph for this function:



8.8.2.5 int cmd_pollrate (usbdevice * *kb*, usbmode * *dummy1*, int *dummy2*, int *rate*, const char * *dummy3*)

Definition at line 52 of file device_mouse.c.

References MSG_SIZE, usbdevice::pollrate, and usbsend.

```
52                                                                                  {
53     uchar msg[MSG_SIZE] = {
54         0x07, 0x0a, 0, 0, (uchar)rate
55     };
56     if(!usbsend(kb, msg, 1))
57         return -1;
58     // Device should disconnect+reconnect, but update the poll rate field in case it doesn't
59     kb->pollrate = rate;
60     return 0;
61 }
```

8.8.2.6 int setactive_kb (usbdevice * *kb*, int *active*)

Definition at line 18 of file device_keyboard.c.

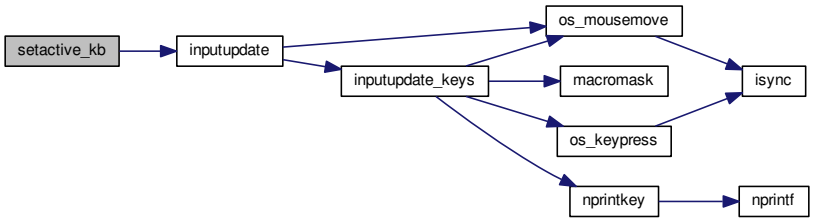
References ACT_LIGHT, ACT_LOCK, ACT_M1, ACT_M2, ACT_M3, ACT_MR_RING, usbdevice::active, DELAY_-MEDIUM, lighting::forceupdate, imutex, IN_CORSAIR, IN_HID, usbdevice::input, inputupdate(), keymap, usbinput-::keys, usbprofile::lastlight, MSG_SIZE, N_KEYS_HW, NEEDS_FW_UPDATE, usbdevice::profile, usbsend, and usbdevice::vtable.

Referenced by cmd_active_kb(), and cmd_idle_kb().

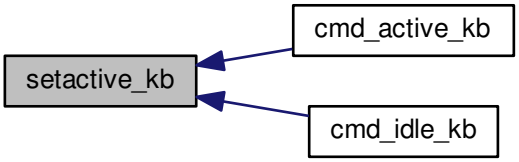
```
18                                          {
19     if(NEEDS_FW_UPDATE(kb))
20         return 0;
21
22     pthread_mutex_lock(imutex(kb));
23     kb->active = !!active;
24     kb->profile->lastlight.forceupdate = 1;
25     // Clear input
26     memset(&kb->input.keys, 0, sizeof(kb->input.keys));
27     inputupdate(kb);
28     pthread_mutex_unlock(imutex(kb));
29
30     uchar msg[3][MSG_SIZE] = {
31         { 0x07, 0x04, 0 },                  // Disables or enables HW control for top row
32         { 0x07, 0x40, 0 },                  // Selects key input
33         { 0x07, 0x05, 2, 0, 0x03, 0x00 }    // Commits key input selection
34     };
35     if(active){
36         // Put the M-keys (K95) as well as the Brightness/Lock keys into software-controlled mode.
37         msg[0][2] = 2;
38         if(!usbsend(kb, msg[0], 1))
39             return -1;
40         DELAY_MEDIUM(kb);
41         // Set input mode on the keys. They must be grouped into packets of 60 bytes (+ 4 bytes header)
42         // Keys are referenced in byte pairs, with the first byte representing the key and the second byte
       representing the mode.
43         for(int key = 0; key < N_KEYS_HW; ){
44             int pair;
45             for(pair = 0; pair < 30 && key < N_KEYS_HW; pair++, key++){
46                 // Select both standard and Corsair input. The standard input will be ignored except in
       BIOS mode.
47                 uchar action = IN_HID | IN_CORSAIR;
48                 // Additionally, make MR activate the MR ring (this is disabled for now, may be back later)
49                 //if(keymap[key].name && !strcmp(keymap[key].name, "mr"))
50                 //    action |= ACT_MR_RING;
51                 msg[1][4 + pair * 2] = key;
52                 msg[1][5 + pair * 2] = action;
53             }
54             // Byte 2 = pair count (usually 30, less on final message)
55             msg[1][2] = pair;
56             if(!usbsend(kb, msg[1], 1))
57                 return -1;
58         }
59         // Commit new input settings
60         if(!usbsend(kb, msg[2], 1))
61             return -1;
62         DELAY_MEDIUM(kb);
63     } else {
64         // Set the M-keys back into hardware mode, restore hardware RGB profile. It has to be sent twice
       for some reason.
65         msg[0][2] = 1;
66         if(!usbsend(kb, msg[0], 1))
67             return -1;
68         DELAY_MEDIUM(kb);
69         if(!usbsend(kb, msg[0], 1))
70             return -1;
71         DELAY_MEDIUM(kb);
72 #ifdef OS_LINUX
73         // On OSX the default key mappings are fine. On Linux, the G keys will freeze the keyboard. Set the
       keyboard entirely to HID input.
74         for(int key = 0; key < N_KEYS_HW; ){
75             int pair;
76             for(pair = 0; pair < 30 && key < N_KEYS_HW; pair++, key++){
77                 uchar action = IN_HID;
78                 // Enable hardware actions
79                 if(keymap[key].name){
80                     if(!strcmp(keymap[key].name, "mr"))
81                         action = ACT_MR_RING;
82                     else if(!strcmp(keymap[key].name, "m1"))
83                         action = ACT_M1;
84                     else if(!strcmp(keymap[key].name, "m2"))
85                         action = ACT_M2;
86                     else if(!strcmp(keymap[key].name, "m3"))
87                         action = ACT_M3;
88                     else if(!strcmp(keymap[key].name, "light"))
89                         action = ACT_LIGHT;
90                     else if(!strcmp(keymap[key].name, "lock"))
91                         action = ACT_LOCK;
```

```
92                 }
93                 msg[1][4 + pair * 2] = key;
94                 msg[1][5 + pair * 2] = action;
95             }
96             // Byte 2 = pair count (usually 30, less on final message)
97             msg[1][2] = pair;
98             if(!usbsend(kb, msg[1], 1))
99                 return -1;
100        }
101        // Commit new input settings
102        if(!usbsend(kb, msg[2], 1))
103            return -1;
104        DELAY_MEDIUM(kb);
105 #endif
106     }
107     // Update indicator LEDs if the profile contains settings for them
108     kb->vtable->updateindicators(kb, 0);
109     return 0;
110 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



8.8.2.7 int setactive_mouse (usbdevice * *kb*, int *active*)

Definition at line 9 of file device_mouse.c.

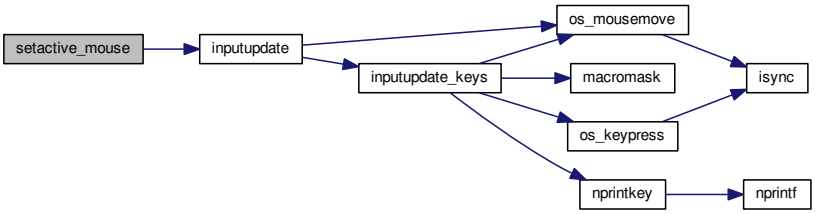
References usbdevice::active, lighting::forceupdate, imutex, IN_CORSAIR, IN_HID, usbdevice::input, inputupdate(), usbinput::keys, usbprofile::lastlight, MSG_SIZE, NEEDS_FW_UPDATE, usbdevice::profile, and usbsend.

Referenced by cmd_active_mouse(), and cmd_idle_mouse().

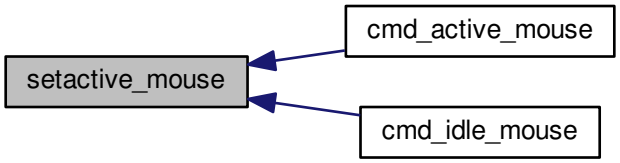
```
9                                           {
10     if(NEEDS_FW_UPDATE(kb))
11         return 0;
12     const int keycount = 20;
13     uchar msg[2][MSG_SIZE] = {
```

```
14          { 0x07, 0x04, 0 },                  // Disables or enables HW control for DPI and Sniper button
15          { 0x07, 0x40, keycount, 0 },        // Select button input (simlilar to the packet sent to
        keyboards, but lacks a commit packet)
16      };
17      if(active)
18          // Put the mouse into SW mode
19          msg[0][2] = 2;
20      else
21          // Restore HW mode
22          msg[0][2] = 1;
23      pthread_mutex_lock(imutex(kb));
24      kb->active = !!active;
25      kb->profile->lastlight.forceupdate = 1;
26      // Clear input
27      memset(&kb->input.keys, 0, sizeof(kb->input.keys));
28      inputupdate(kb);
29      pthread_mutex_unlock(imutex(kb));
30      if(!usbsend(kb, msg[0], 1))
31          return -1;
32      if(active){
33          // Set up key input
34          if(!usbsend(kb, msg[1], 1))
35              return -1;
36          for(int i = 0; i < keycount; i++){
37              msg[1][i * 2 + 4] = i + 1;
38              msg[1][i * 2 + 5] = (i < 6 ? IN_HID : IN_CORSAIR);
39          }
40      }
41      return 0;
42  }
```

Here is the call graph for this function:



Here is the caller graph for this function:



8.8.2.8 void setmodeindex_nrgb (usbdevice * *kb*, int *index*)

Definition at line 120 of file device_keyboard.c.

References NK95_M1, NK95_M2, NK95_M3, and nk95cmd.

```
120                                                 {
121     switch(index % 3){
122     case 0:
123         nk95cmd(kb, NK95_M1);
124         break;
125     case 1:
126         nk95cmd(kb, NK95_M2);
127         break;
128     case 2:
129         nk95cmd(kb, NK95_M3);
130         break;
131     }
132 }
```

8.8.2.9 int start_dev (usbdevice * kb, int makeactive)

Definition at line 55 of file device.c.

References _start_dev(), USB_DELAY_DEFAULT, and usbdevice::usbdelay.

```
55                                                  {
56     // Force USB interval to 10ms during initial setup phase; return to nominal 5ms after setup completes.
57     kb->usbdelay = 10;
58     int res = _start_dev(kb, makeactive);
59     kb->usbdelay = USB_DELAY_DEFAULT;
60     return res;
61 }
```

Here is the call graph for this function:

8.8.2.10 int start_kb_nrgb (usbdevice * kb, int makeactive)

Definition at line 9 of file device_keyboard.c.

References usbdevice::active, NK95_HWOFF, nk95cmd, and usbdevice::pollrate.

```
9                                                   {
10     // Put the non-RGB K95 into software mode. Nothing else needs to be done hardware wise
11     nk95cmd(kb, NK95_HWOFF);
12     // Fill out RGB features for consistency, even though the keyboard doesn't have them
13     kb->active = 1;
14     kb->pollrate = -1;
15     return 0;
16 }
```

8.8.3 Variable Documentation

8.8.3.1 pthread_mutex_t devmutex[9]

Definition at line 12 of file device.c.

Referenced by _updateconnected(), quitWithLock(), and usb_rm_device().

8.8.3.2 pthread_mutex_t inputmutex[9]

Definition at line 13 of file device.c.

8.8.3.3 usbdevice keyboard[9]

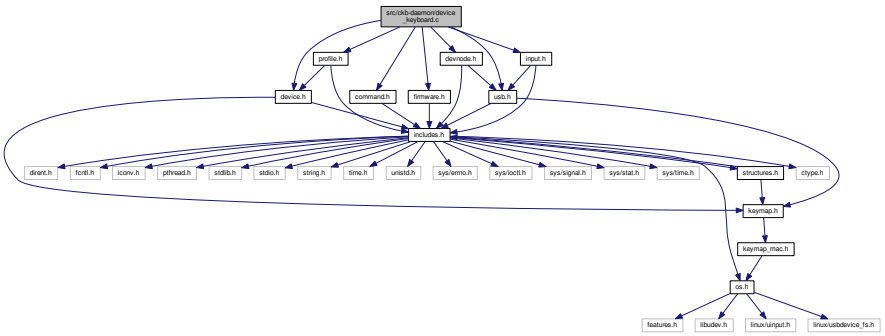
Definition at line 10 of file device.c.

Referenced by _mkdevpath(), _mknotifynode(), _rmnotifynode(), _setupusb(), _updateconnected(), closeusb(), main(), mkfwnode(), os_closeusb(), os_inputmain(), os_inputopen(), os_setupusb(), quitWithLock(), rmdevpath(), usb_rm_device(), and usbadd().

8.9 src/ckb-daemon/device_keyboard.c File Reference

```
#include "command.h"
#include "device.h"
#include "devnode.h"
#include "firmware.h"
#include "input.h"
#include "profile.h"
#include "usb.h"
```

Include dependency graph for device_keyboard.c:



Functions

- int start_kb_nrgb (usbdevice *kb, int makeactive)
- int setactive_kb (usbdevice *kb, int active)
- int cmd_active_kb (usbdevice *kb, usbmode *dummy1, int dummy2, int dummy3, const char *dummy4)
- int cmd_idle_kb (usbdevice *kb, usbmode *dummy1, int dummy2, int dummy3, const char *dummy4)
- void setmodeindex_nrgb (usbdevice *kb, int index)

8.9.1 Function Documentation

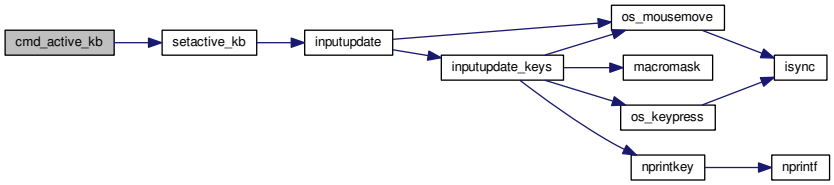
8.9.1.1 int cmd_active_kb (usbdevice * kb, usbmode * dummy1, int dummy2, int dummy3, const char * dummy4)

Definition at line 112 of file device_keyboard.c.

References setactive_kb().

```
112                                                                                                 {
113     return setactive_kb(kb, 1);
114 }
```

Here is the call graph for this function:



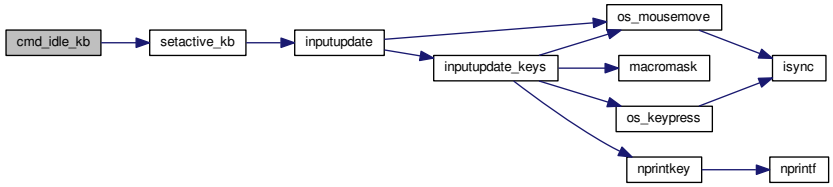
8.9.1.2 int cmd_idle_kb (usbdevice * *kb*, usbmode * *dummy1*, int *dummy2*, int *dummy3*, const char * *dummy4*)

Definition at line 116 of file device_keyboard.c.

References setactive_kb().

```
116                                                                                 {
117     return setactive_kb(kb, 0);
118 }
```

Here is the call graph for this function:



8.9.1.3 int setactive_kb (usbdevice * *kb*, int *active*)

Definition at line 18 of file device_keyboard.c.

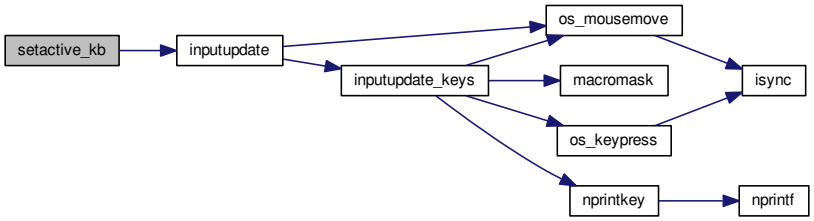
References ACT_LIGHT, ACT_LOCK, ACT_M1, ACT_M2, ACT_M3, ACT_MR_RING, usbdevice::active, DELAY_-MEDIUM, lighting::forceupdate, imutex, IN_CORSAIR, IN_HID, usbdevice::input, inputupdate(), keymap, usbinput-::keys, usbprofile::lastlight, MSG_SIZE, N_KEYS_HW, NEEDS_FW_UPDATE, usbdevice::profile, usbsend, and usbdevice::vtable.

Referenced by cmd_active_kb(), and cmd_idle_kb().

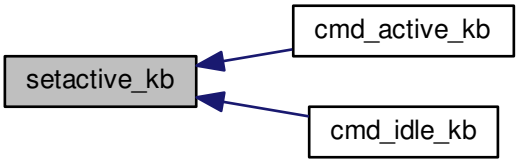
```
18                                          {
19     if(NEEDS_FW_UPDATE(kb))
20         return 0;
21
22     pthread_mutex_lock(imutex(kb));
23     kb->active = !!active;
24     kb->profile->lastlight.forceupdate = 1;
25     // Clear input
26     memset(&kb->input.keys, 0, sizeof(kb->input.keys));
27     inputupdate(kb);
28     pthread_mutex_unlock(imutex(kb));
29
30     uchar msg[3][MSG_SIZE] = {
31         { 0x07, 0x04, 0 },                  // Disables or enables HW control for top row
32         { 0x07, 0x40, 0 },                  // Selects key input
```

```
33          { 0x07, 0x05, 2, 0, 0x03, 0x00 }    // Commits key input selection
34      };
35      if(active){
36          // Put the M-keys (K95) as well as the Brightness/Lock keys into software-controlled mode.
37          msg[0][2] = 2;
38          if(!usbsend(kb, msg[0], 1))
39              return -1;
40          DELAY_MEDIUM(kb);
41          // Set input mode on the keys. They must be grouped into packets of 60 bytes (+ 4 bytes header)
42          // Keys are referenced in byte pairs, with the first byte representing the key and the second byte
        representing the mode.
43          for(int key = 0; key < N_KEYS_HW; ){
44              int pair;
45              for(pair = 0; pair < 30 && key < N_KEYS_HW; pair++, key++){
46                  // Select both standard and Corsair input. The standard input will be ignored except in
        BIOS mode.
47                  uchar action = IN_HID | IN_CORSAIR;
48                  // Additionally, make MR activate the MR ring (this is disabled for now, may be back later)
49                  //if(keymap[key].name && !strcmp(keymap[key].name, "mr"))
50                  //    action |= ACT_MR_RING;
51                  msg[1][4 + pair * 2] = key;
52                  msg[1][5 + pair * 2] = action;
53              }
54              // Byte 2 = pair count (usually 30, less on final message)
55              msg[1][2] = pair;
56              if(!usbsend(kb, msg[1], 1))
57                  return -1;
58          }
59          // Commit new input settings
60          if(!usbsend(kb, msg[2], 1))
61              return -1;
62          DELAY_MEDIUM(kb);
63      } else {
64          // Set the M-keys back into hardware mode, restore hardware RGB profile. It has to be sent twice
        for some reason.
65          msg[0][2] = 1;
66          if(!usbsend(kb, msg[0], 1))
67              return -1;
68          DELAY_MEDIUM(kb);
69          if(!usbsend(kb, msg[0], 1))
70              return -1;
71          DELAY_MEDIUM(kb);
72  #ifdef OS_LINUX
73          // On OSX the default key mappings are fine. On Linux, the G keys will freeze the keyboard. Set the
        keyboard entirely to HID input.
74          for(int key = 0; key < N_KEYS_HW; ){
75              int pair;
76              for(pair = 0; pair < 30 && key < N_KEYS_HW; pair++, key++){
77                  uchar action = IN_HID;
78                  // Enable hardware actions
79                  if(keymap[key].name){
80                      if(!strcmp(keymap[key].name, "mr"))
81                          action = ACT_MR_RING;
82                      else if(!strcmp(keymap[key].name, "m1"))
83                          action = ACT_M1;
84                      else if(!strcmp(keymap[key].name, "m2"))
85                          action = ACT_M2;
86                      else if(!strcmp(keymap[key].name, "m3"))
87                          action = ACT_M3;
88                      else if(!strcmp(keymap[key].name, "light"))
89                          action = ACT_LIGHT;
90                      else if(!strcmp(keymap[key].name, "lock"))
91                          action = ACT_LOCK;
92                  }
93                  msg[1][4 + pair * 2] = key;
94                  msg[1][5 + pair * 2] = action;
95              }
96              // Byte 2 = pair count (usually 30, less on final message)
97              msg[1][2] = pair;
98              if(!usbsend(kb, msg[1], 1))
99                  return -1;
100         }
101         // Commit new input settings
102         if(!usbsend(kb, msg[2], 1))
103             return -1;
104         DELAY_MEDIUM(kb);
105 #endif
106     }
107     // Update indicator LEDs if the profile contains settings for them
108     kb->vtable->updateindicators(kb, 0);
109     return 0;
110 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



8.9.1.4 void setmodeindex_nrgb (usbdevice * *kb*, int *index*)

Definition at line 120 of file device_keyboard.c.

References NK95_M1, NK95_M2, NK95_M3, and nk95cmd.

```
120                                                 {
121     switch(index % 3){
122     case 0:
123         nk95cmd(kb, NK95_M1);
124         break;
125     case 1:
126         nk95cmd(kb, NK95_M2);
127         break;
128     case 2:
129         nk95cmd(kb, NK95_M3);
130         break;
131     }
132 }
```

8.9.1.5 int start_kb_nrgb (usbdevice * *kb*, int *makeactive*)

Definition at line 9 of file device_keyboard.c.

References usbdevice::active, NK95_HWOFF, nk95cmd, and usbdevice::pollrate.

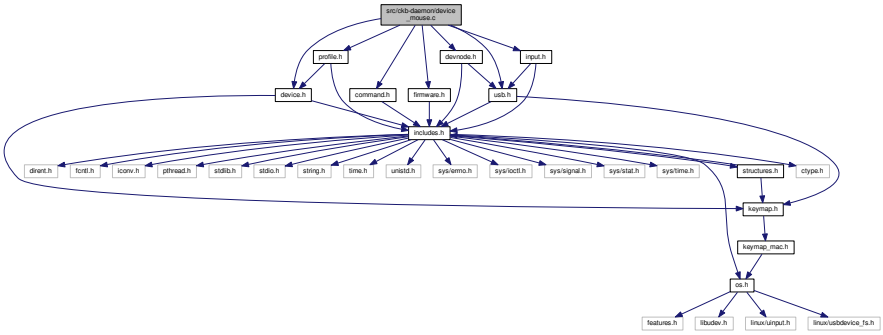
```
9                                                   {
10     // Put the non-RGB K95 into software mode. Nothing else needs to be done hardware wise
11     nk95cmd(kb, NK95_HWOFF);
12     // Fill out RGB features for consistency, even though the keyboard doesn't have them
13     kb->active = 1;
```

```
14     kb->pollrate = -1;
15     return 0;
16 }
```

8.10 src/ckb-daemon/device_mouse.c File Reference

```
#include "command.h"
#include "device.h"
#include "devnode.h"
#include "firmware.h"
#include "input.h"
#include "profile.h"
#include "usb.h"
```

Include dependency graph for device_mouse.c:



Functions

- int setactive_mouse (usbdevice ∗kb, int active)
- int cmd_active_mouse (usbdevice ∗kb, usbmode ∗dummy1, int dummy2, int dummy3, const char ∗dummy4)
- int cmd_idle_mouse (usbdevice ∗kb, usbmode ∗dummy1, int dummy2, int dummy3, const char ∗dummy4)
- int cmd_pollrate (usbdevice ∗kb, usbmode ∗dummy1, int dummy2, int rate, const char ∗dummy3)

8.10.1 Function Documentation

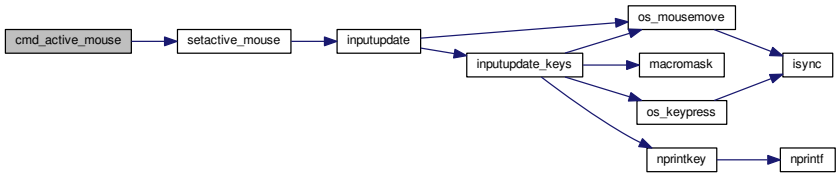
8.10.1.1 int cmd_active_mouse (usbdevice ∗ *kb,* usbmode ∗ *dummy1,* int *dummy2,* int *dummy3,* const char ∗ *dummy4*)

Definition at line 44 of file device_mouse.c.

References setactive_mouse().

```
44                                                                                  {
45     return setactive_mouse(kb, 1);
46 }
```

Here is the call graph for this function:



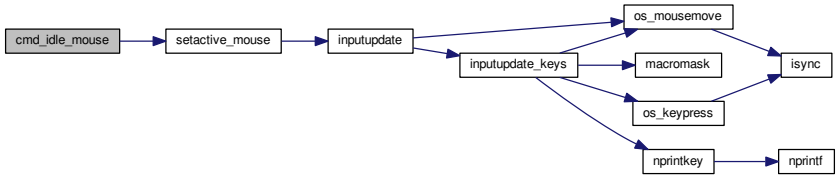
8.10.1.2 int cmd_idle_mouse (usbdevice * *kb*, usbmode * *dummy1*, int *dummy2*, int *dummy3*, const char * *dummy4*)

Definition at line 48 of file device_mouse.c.

References setactive_mouse().

```
48                                                                                                  {
49     return setactive_mouse(kb, 0);
50 }
```

Here is the call graph for this function:



8.10.1.3 int cmd_pollrate (usbdevice * *kb*, usbmode * *dummy1*, int *dummy2*, int *rate*, const char * *dummy3*)

Definition at line 52 of file device_mouse.c.

References MSG_SIZE, usbdevice::pollrate, and usbsend.

```
52                                                                                                  {
53     uchar msg[MSG_SIZE] = {
54         0x07, 0x0a, 0, 0, (uchar)rate
55     };
56     if(!usbsend(kb, msg, 1))
57         return -1;
58     // Device should disconnect+reconnect, but update the poll rate field in case it doesn't
59     kb->pollrate = rate;
60     return 0;
61 }
```

8.10.1.4 int setactive_mouse (usbdevice * *kb*, int *active*)

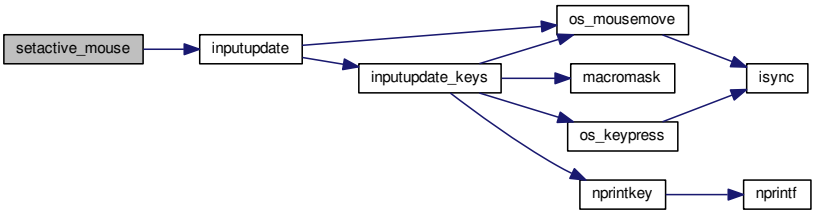
Definition at line 9 of file device_mouse.c.

References usbdevice::active, lighting::forceupdate, imutex, IN_CORSAIR, IN_HID, usbdevice::input, inputupdate(), usbinput::keys, usbprofile::lastlight, MSG_SIZE, NEEDS_FW_UPDATE, usbdevice::profile, and usbsend.

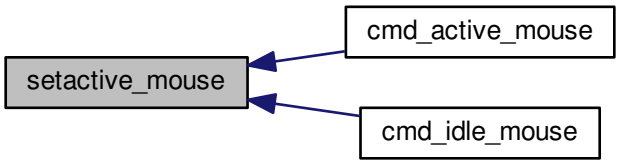
Referenced by cmd_active_mouse(), and cmd_idle_mouse().

```
9                                                   {
10      if(NEEDS_FW_UPDATE(kb))
11          return 0;
12      const int keycount = 20;
13      uchar msg[2][MSG_SIZE] = {
14          { 0x07, 0x04, 0 },                  // Disables or enables HW control for DPI and Sniper button
15          { 0x07, 0x40, keycount, 0 },        // Select button input (simlilar to the packet sent to
        keyboards, but lacks a commit packet)
16      };
17      if(active)
18          // Put the mouse into SW mode
19          msg[0][2] = 2;
20      else
21          // Restore HW mode
22          msg[0][2] = 1;
23      pthread_mutex_lock(imutex(kb));
24      kb->active = !!active;
25      kb->profile->lastlight.forceupdate = 1;
26      // Clear input
27      memset(&kb->input.keys, 0, sizeof(kb->input.keys));
28      inputupdate(kb);
29      pthread_mutex_unlock(imutex(kb));
30      if(!usbsend(kb, msg[0], 1))
31          return -1;
32      if(active){
33          // Set up key input
34          if(!usbsend(kb, msg[1], 1))
35              return -1;
36          for(int i = 0; i < keycount; i++){
37              msg[1][i * 2 + 4] = i + 1;
38              msg[1][i * 2 + 5] = (i < 6 ? IN_HID : IN_CORSAIR);
39          }
40      }
41      return 0;
42  }
```

Here is the call graph for this function:



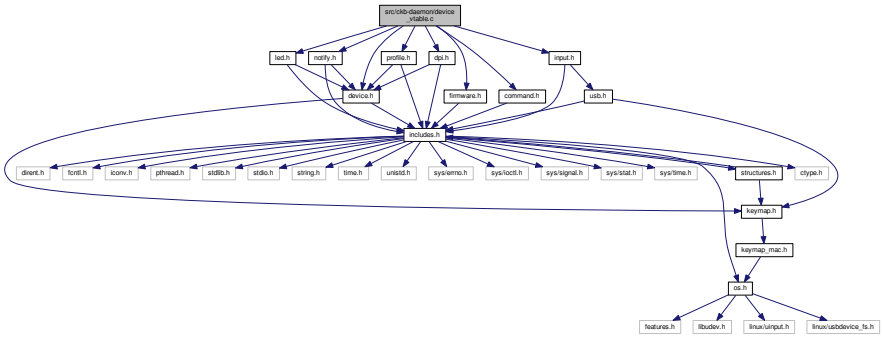
Here is the caller graph for this function:



8.11 src/ckb-daemon/device_vtable.c File Reference

```
#include "command.h"
#include "device.h"
#include "dpi.h"
#include "firmware.h"
#include "input.h"
#include "led.h"
#include "notify.h"
#include "profile.h"
```

Include dependency graph for device_vtable.c:



Functions

- static void cmd_none (usbdevice ∗kb, usbmode ∗dummy1, int dummy2, int dummy3, const char ∗dummy4)
- static int cmd_io_none (usbdevice ∗kb, usbmode ∗dummy1, int dummy2, int dummy3, const char ∗dummy4)
- static void cmd_macro_none (usbdevice ∗kb, usbmode ∗dummy1, int dummy2, const char ∗dummy3, const char ∗dummy4)
- static int loadprofile_none (usbdevice ∗kb)
- static void int1_void_none (usbdevice ∗kb, int dummy)
- static int int1_int_none (usbdevice ∗kb, int dummy)

Variables

- const devcmd vtable_keyboard
- const devcmd vtable_keyboard_nonrgb
- const devcmd vtable_mouse

8.11.1 Function Documentation

8.11.1.1 static int cmd_io_none (usbdevice ∗ *kb,* usbmode ∗ *dummy1,* int *dummy2,* int *dummy3,* const char ∗ *dummy4*) [static]

Definition at line 13 of file device_vtable.c.

```
13                                                                              {
14     return 0;
15 }
```

8.11.1.2 static void cmd_macro_none (usbdevice * kb, usbmode * dummy1, int dummy2, const char * dummy3, const char * dummy4) [static]

Definition at line 16 of file device_vtable.c.

```
16
        {
17 }
```

8.11.1.3 static void cmd_none (usbdevice * kb, usbmode * dummy1, int dummy2, int dummy3, const char * dummy4) [static]

Definition at line 11 of file device_vtable.c.

```
11                                                                                                  {
12 }
```

8.11.1.4 static int int1_int_none (usbdevice * kb, int dummy) [static]

Definition at line 23 of file device_vtable.c.

```
23                                                  {
24     return 0;
25 }
```

8.11.1.5 static void int1_void_none (usbdevice * kb, int dummy) [static]

Definition at line 21 of file device_vtable.c.

```
21                                                    {
22 }
```

8.11.1.6 static int loadprofile_none (usbdevice * kb) [static]

Definition at line 18 of file device_vtable.c.

```
18                                           {
19     return 0;
20 }
```

8.11.2 Variable Documentation

8.11.2.1 const devcmd vtable_keyboard

Definition at line 28 of file device_vtable.c.

Referenced by get_vtable().

8.11.2.2 const devcmd vtable_keyboard_nonrgb

Definition at line 75 of file device_vtable.c.

Referenced by get_vtable().

8.11.2.3 const devcmd vtable_mouse

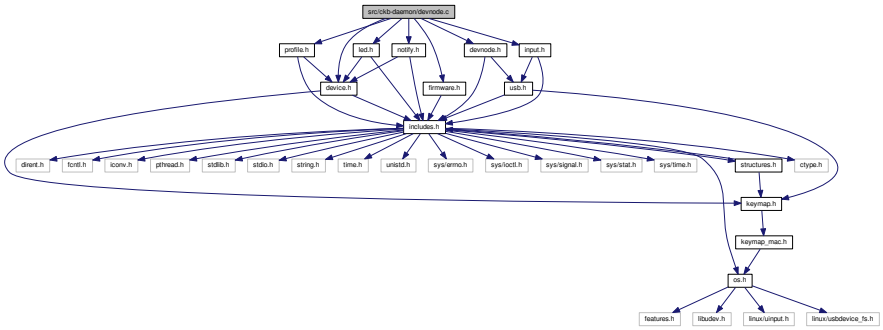
Definition at line 122 of file device_vtable.c.

Referenced by get_vtable().

8.12 src/ckb-daemon/devnode.c File Reference

```
#include "device.h"
#include "devnode.h"
#include "firmware.h"
#include "input.h"
#include "led.h"
#include "notify.h"
#include "profile.h"
```

Include dependency graph for devnode.c:



Data Structures

- struct _readlines_ctx

Macros

- #define S_GID_READ (gid >= 0 ? S_CUSTOM_R : S_READ)
- #define MAX_BUFFER (1024 * 1024 - 1)

Functions

- int rm_recursive (const char *path)
- void _updateconnected ()
- void updateconnected ()
- int _mknotifynode (usbdevice *kb, int notify)
- int mknotifynode (usbdevice *kb, int notify)
- int _rmnotifynode (usbdevice *kb, int notify)
- int rmnotifynode (usbdevice *kb, int notify)
- static int _mkdevpath (usbdevice *kb)
- int mkdevpath (usbdevice *kb)
- int rmdevpath (usbdevice *kb)
- int mkfwnode (usbdevice *kb)
- void readlines_ctx_init (readlines_ctx *ctx)

- void readlines_ctx_free (readlines_ctx ctx)
- unsigned readlines (int fd, readlines_ctx ctx, const char **input)

Variables

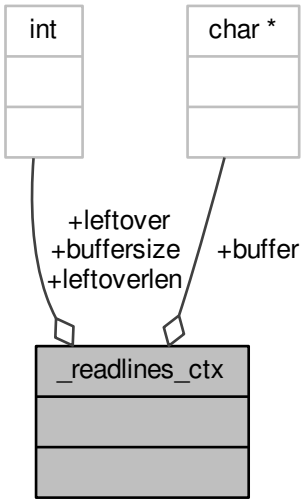
- const char *const devpath = "/dev/input/ckb"
- long gid = -1

8.12.1 Data Structure Documentation

8.12.1.1 struct _readlines_ctx

Definition at line 318 of file devnode.c.

Collaboration diagram for _readlines_ctx:



Data Fields

char *	buffer	
int	buffersize	
int	leftover	
int	leftoverlen	

8.12.2 Macro Definition Documentation

8.12.2.1 #define MAX_BUFFER (1024 * 1024 - 1)

Definition at line 317 of file devnode.c.

Referenced by readlines().

8.12.2.2 #define S_GID_READ (gid >= 0 ? S_CUSTOM_R : S_READ)

Definition at line 17 of file devnode.c.

Referenced by _mkdevpath(), _mknotifynode(), _updateconnected(), and mkfwnode().

8.12.3 Function Documentation

8.12.3.1 static int _mkdevpath (usbdevice * kb) [static]

Definition at line 119 of file devnode.c.

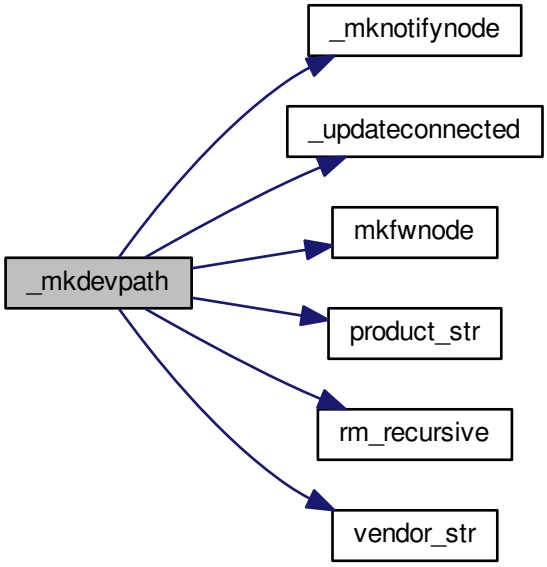
References _mknotifynode(), _updateconnected(), ckb_err, ckb_warn, devpath, FEAT_ADJRATE, FEAT_BIND, FEAT_FWUPDATE, FEAT_FWVERSION, FEAT_MONOCHROME, FEAT_NOTIFY, FEAT_POLLRATE, FEAT_RGB, gid, HAS_FEATURES, INDEX_OF, usbdevice::infifo, keyboard, mkfwnode(), usbdevice::name, usbdevice::product, product_str(), rm_recursive(), S_CUSTOM, S_GID_READ, S_READ, S_READDIR, S_READWRITE, usbdevice::serial, usbdevice::vendor, and vendor_str().

Referenced by mkdevpath().

```
119                                 {
120     int index = INDEX_OF(kb, keyboard);
121     // Create the control path
122     char path[strlen(devpath) + 2];
123     snprintf(path, sizeof(path), "%s%d", devpath, index);
124     if(rm_recursive(path) != 0 && errno != ENOENT){
125         ckb_err("Unable to delete %s: %s\n", path, strerror(errno));
126         return -1;
127     }
128     if(mkdir(path, S_READDIR) != 0){
129         ckb_err("Unable to create %s: %s\n", path, strerror(errno));
130         rm_recursive(path);
131         return -1;
132     }
133     if(gid >= 0)
134         chown(path, 0, gid);
135 
136     if(kb == keyboard + 0){
137         // Root keyboard: write a list of devices
138         _updateconnected();
139         // Write version number
140         char vpath[sizeof(path) + 8];
141         snprintf(vpath, sizeof(vpath), "%s/version", path);
142         FILE* vfile = fopen(vpath, "w");
143         if(vfile){
144             fprintf(vfile, "%s\n", CKB_VERSION_STR);
145             fclose(vfile);
146             chmod(vpath, S_GID_READ);
147             if(gid >= 0)
148                 chown(vpath, 0, gid);
149         } else {
150             ckb_warn("Unable to create %s: %s\n", vpath, strerror(errno));
151             remove(vpath);
152         }
153         // Write PID
154         char ppath[sizeof(path) + 4];
155         snprintf(ppath, sizeof(ppath), "%s/pid", path);
156         FILE* pfile = fopen(ppath, "w");
157         if(pfile){
158             fprintf(pfile, "%u\n", getpid());
159             fclose(pfile);
160             chmod(ppath, S_READ);
161             if(gid >= 0)
162                 chown(vpath, 0, gid);
163         } else {
164             ckb_warn("Unable to create %s: %s\n", ppath, strerror(errno));
165             remove(ppath);
166         }
167     } else {
168         // Create command FIFO
169         char inpath[sizeof(path) + 4];
170         snprintf(inpath, sizeof(inpath), "%s/cmd", path);
171         if(mkfifo(inpath, gid >= 0 ? S_CUSTOM : S_READWRITE) != 0
172                 // Open the node in RDWR mode because RDONLY will lock the thread
173                 || (kb->infifo = open(inpath, O_RDWR) + 1) == 0){
174             // Add one to the FD because 0 is a valid descriptor, but ckb uses 0 for uninitialized devices
175             ckb_err("Unable to create %s: %s\n", inpath, strerror(errno));
176             rm_recursive(path);
```

```
177             kb->infifo = 0;
178             return -1;
179         }
180         if(gid >= 0)
181             fchown(kb->infifo - 1, 0, gid);
182 
183         // Create notification FIFO
184         _mknotifynode(kb, 0);
185 
186         // Write the model and serial to files
187         char mpath[sizeof(path) + 6], spath[sizeof(path) + 7];
188         snprintf(mpath, sizeof(mpath), "%s/model", path);
189         snprintf(spath, sizeof(spath), "%s/serial", path);
190         FILE* mfile = fopen(mpath, "w");
191         if(mfile){
192             fputs(kb->name, mfile);
193             fputc('\n', mfile);
194             fclose(mfile);
195             chmod(mpath, S_GID_READ);
196             if(gid >= 0)
197                 chown(mpath, 0, gid);
198         } else {
199             ckb_warn("Unable to create %s: %s\n", mpath, strerror(errno));
200             remove(mpath);
201         }
202         FILE* sfile = fopen(spath, "w");
203         if(sfile){
204             fputs(kb->serial, sfile);
205             fputc('\n', sfile);
206             fclose(sfile);
207             chmod(spath, S_GID_READ);
208             if(gid >= 0)
209                 chown(spath, 0, gid);
210         } else {
211             ckb_warn("Unable to create %s: %s\n", spath, strerror(errno));
212             remove(spath);
213         }
214         // Write the keyboard's features
215         char fpath[sizeof(path) + 9];
216         snprintf(fpath, sizeof(fpath), "%s/features", path);
217         FILE* ffile = fopen(fpath, "w");
218         if(ffile){
219             fprintf(ffile, "%s %s", vendor_str(kb->vendor),
    product_str(kb->product));
220             if(HAS_FEATURES(kb, FEAT_MONOCHROME))
221                 fputs(" monochrome", ffile);
222             if(HAS_FEATURES(kb, FEAT_RGB))
223                 fputs(" rgb", ffile);
224             if(HAS_FEATURES(kb, FEAT_POLLRATE))
225                 fputs(" pollrate", ffile);
226             if(HAS_FEATURES(kb, FEAT_ADJRATE))
227                 fputs(" adjrate", ffile);
228             if(HAS_FEATURES(kb, FEAT_BIND))
229                 fputs(" bind", ffile);
230             if(HAS_FEATURES(kb, FEAT_NOTIFY))
231                 fputs(" notify", ffile);
232             if(HAS_FEATURES(kb, FEAT_FWVERSION))
233                 fputs(" fwversion", ffile);
234             if(HAS_FEATURES(kb, FEAT_FWUPDATE))
235                 fputs(" fwupdate", ffile);
236             fputc('\n', ffile);
237             fclose(ffile);
238             chmod(fpath, S_GID_READ);
239             if(gid >= 0)
240                 chown(fpath, 0, gid);
241         } else {
242             ckb_warn("Unable to create %s: %s\n", fpath, strerror(errno));
243             remove(fpath);
244         }
245         // Write firmware version and poll rate
246         mkfwnode(kb);
247     }
248     return 0;
249 }
```

Here is the call graph for this function:



Here is the caller graph for this function:

8.12.3.2 int _mknotifynode (usbdevice * *kb,* int *notify*)

Definition at line 70 of file devnode.c.

References ckb_warn, devpath, gid, INDEX_OF, keyboard, usbdevice::outfifo, OUTFIFO_MAX, and S_GID_READ.

Referenced by _mkdevpath(), and mknotifynode().

```
70                                          {
71     if(notify < 0 || notify >= OUTFIFO_MAX)
72         return -1;
73     if(kb->outfifo[notify] != 0)
74         return 0;
75     // Create the notification node
76     int index = INDEX_OF(kb, keyboard);
77     char outpath[strlen(devpath) + 10];
78     snprintf(outpath, sizeof(outpath), "%s%d/notify%d", devpath, index, notify);
79     if(mkfifo(outpath, S_GID_READ) != 0 || (kb->outfifo[notify] = open(outpath, O_RDWR |
       O_NONBLOCK) + 1) == 0){
80         // Add one to the FD because 0 is a valid descriptor, but ckb uses 0 for uninitialized devices
81         ckb_warn("Unable to create %s: %s\n", outpath, strerror(errno));
82         kb->outfifo[notify] = 0;
83         remove(outpath);
84         return -1;
85     }
86     if(gid >= 0)
87         fchown(kb->outfifo[notify] - 1, 0, gid);
88     return 0;
89 }
```

Here is the caller graph for this function:

8.12.3.3 int _rmnotifynode (usbdevice * kb, int notify)

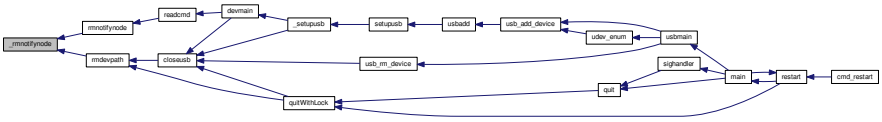
Definition at line 98 of file devnode.c.

References devpath, INDEX_OF, keyboard, usbdevice::outfifo, and OUTFIFO_MAX.

Referenced by rmdevpath(), and rmnotifynode().

```
98                                              {
99      if(notify < 0 || notify >= OUTFIFO_MAX || !kb->outfifo[notify])
100         return -1;
101      int index = INDEX_OF(kb, keyboard);
102      char outpath[strlen(devpath) + 10];
103      snprintf(outpath, sizeof(outpath), "%s%d/notify%d", devpath, index, notify);
104      // Close FIFO
105      close(kb->outfifo[notify] - 1);
106      kb->outfifo[notify] = 0;
107      // Delete node
108      int res = remove(outpath);
109      return res;
110  }
```

Here is the caller graph for this function:



8.12.3.4 void _updateconnected ( )

Definition at line 38 of file devnode.c.

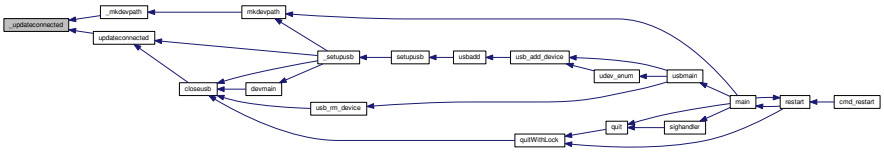
References ckb_warn, DEV_MAX, devmutex, devpath, gid, IS_CONNECTED, keyboard, and S_GID_READ.

Referenced by _mkdevpath(), and updateconnected().

```
38                          {
39      pthread_mutex_lock(devmutex);
40      char cpath[strlen(devpath) + 12];
41      snprintf(cpath, sizeof(cpath), "%s0/connected", devpath);
42      FILE* cfile = fopen(cpath, "w");
43      if(!cfile){
44          ckb_warn("Unable to update %s: %s\n", cpath, strerror(errno));
45          pthread_mutex_unlock(devmutex);
46          return;
47      }
48      int written = 0;
49      for(int i = 1; i < DEV_MAX; i++){
50          if(IS_CONNECTED(keyboard + i)){
51              written = 1;
52              fprintf(cfile, "%s%d %s %s\n", devpath, i, keyboard[i].serial,
    keyboard[i].name);
53          }
54      }
55      if(!written)
56          fputc('\n', cfile);
57      fclose(cfile);
```

```
58     chmod(cpath, S_GID_READ);
59     if(gid >= 0)
60         chown(cpath, 0, gid);
61     pthread_mutex_unlock(devmutex);
62 }
```

Here is the caller graph for this function:



8.12.3.5 int mkdevpath (usbdevice * *kb*)

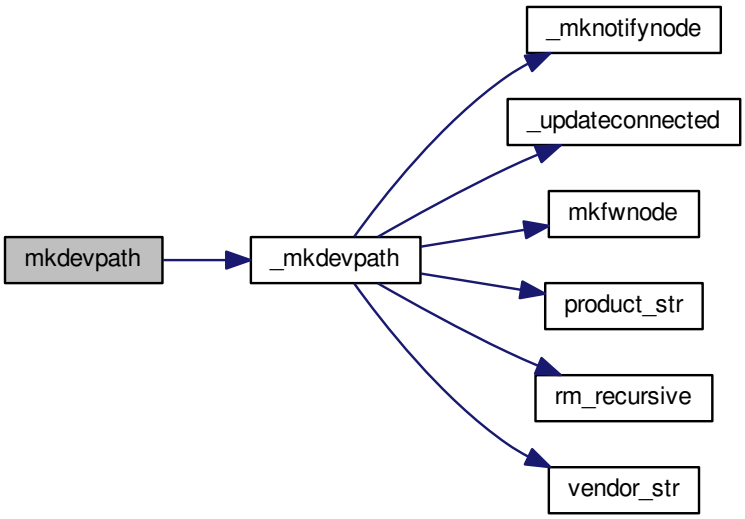
Definition at line 251 of file devnode.c.

References _mkdevpath(), euid_guard_start, and euid_guard_stop.

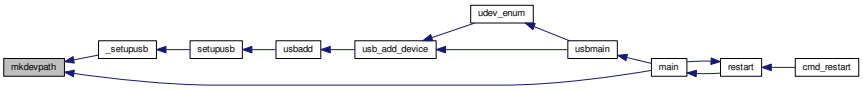
Referenced by _setupusb(), and main().

```
251                                {
252     euid_guard_start;
253     int res = _mkdevpath(kb);
254     euid_guard_stop;
255     return res;
256 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



8.12.3.6 int mkfwnode (usbdevice * kb)

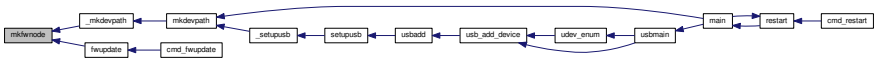
Definition at line 282 of file devnode.c.

References ckb_warn, devpath, usbdevice::fwversion, gid, INDEX_OF, keyboard, usbdevice::pollrate, and S_GID_-READ.

Referenced by _mkdevpath(), and fwupdate().

```
282                             {
283     int index = INDEX_OF(kb, keyboard);
284     char fwpath[strlen(devpath) + 12];
285     snprintf(fwpath, sizeof(fwpath), "%s%d/fwversion", devpath, index);
286     FILE* fwfile = fopen(fwpath, "w");
287     if(fwfile){
288         fprintf(fwfile, "%04x", kb->fwversion);
289         fputc('\n', fwfile);
290         fclose(fwfile);
291         chmod(fwpath, S_GID_READ);
292         if(gid >= 0)
293             chown(fwpath, 0, gid);
294     } else {
295         ckb_warn("Unable to create %s: %s\n", fwpath, strerror(errno));
296         remove(fwpath);
297         return -1;
298     }
299     char ppath[strlen(devpath) + 11];
300     snprintf(ppath, sizeof(ppath), "%s%d/pollrate", devpath, index);
301     FILE* pfile = fopen(ppath, "w");
302     if(pfile){
303         fprintf(pfile, "%d ms", kb->pollrate);
304         fputc('\n', pfile);
305         fclose(pfile);
306         chmod(ppath, S_GID_READ);
307         if(gid >= 0)
308             chown(ppath, 0, gid);
309     } else {
310         ckb_warn("Unable to create %s: %s\n", fwpath, strerror(errno));
311         remove(ppath);
312         return -2;
313     }
314     return 0;
315 }
```

Here is the caller graph for this function:



8.12.3.7 int mknotifynode (usbdevice * kb, int notify)

Definition at line 91 of file devnode.c.

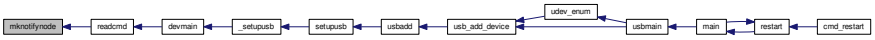
References _mknotifynode(), euid_guard_start, and euid_guard_stop.

Referenced by readcmd().

```
91                                          {
92     euid_guard_start;
93     int res = _mknotifynode(kb, notify);
94     euid_guard_stop;
95     return res;
96 }
```

Here is the call graph for this function:

Here is the caller graph for this function:



8.12.3.8 unsigned readlines (int *fd,* readlines_ctx *ctx,* const char ** *input*)

Definition at line 336 of file devnode.c.

References _readlines_ctx::buffer, _readlines_ctx::buffersize, ckb_warn, _readlines_ctx::leftover, _readlines_ctx::leftoverlen, and MAX_BUFFER.

Referenced by devmain().

```
336                                                                 {
337     // Move any data left over from a previous read to the start of the buffer
338     char* buffer = ctx->buffer;
339     int buffersize = ctx->buffersize;
340     int leftover = ctx->leftover, leftoverlen = ctx->leftoverlen;
341     memcpy(buffer, buffer + leftover, leftoverlen);
342     // Read data from the file
343     ssize_t length = read(fd, buffer + leftoverlen, buffersize - leftoverlen);
344     length = (length < 0 ? 0 : length) + leftoverlen;
345     leftover = ctx->leftover = leftoverlen = ctx->leftoverlen = 0;
346     if(length <= 0){
347         *input = 0;
348         return 0;
349     }
350     // Continue buffering until all available input is read or there's no room left
351     while(length == buffersize){
352         if(buffersize == MAX_BUFFER)
353             break;
354         int oldsize = buffersize;
355         buffersize += 4096;
356         ctx->buffersize = buffersize;
357         buffer = ctx->buffer = realloc(buffer, buffersize + 1);
358         ssize_t length2 = read(fd, buffer + oldsize, buffersize - oldsize);
359         if(length2 <= 0)
360             break;
361         length += length2;
362     }
363     buffer[length] = 0;
364     // Input should be issued one line at a time and should end with a newline.
365     char* lastline = memrchr(buffer, '\n', length);
366     if(lastline == buffer + length - 1){
367         // If the buffer ends in a newline, process the whole string
368         *input = buffer;
369         return length;
370     } else if(lastline){
```

```
371         // Otherwise, chop off the last line but process everything else
372         *lastline = 0;
373         leftover = ctx->leftover = lastline + 1 - buffer;
374         leftoverlen = ctx->leftoverlen = length - leftover;
375         *input = buffer;
376         return leftover - 1;
377     } else {
378         // If a newline wasn't found at all, process the whole buffer next time
379         *input = 0;
380         if(length == MAX_BUFFER){
381             // Unless the buffer is completely full, in which case discard it
382             ckb_warn("Too much input (1MB). Dropping.\n");
383             return 0;
384         }
385         leftoverlen = ctx->leftoverlen = length;
386         return 0;
387     }
388 }
```

Here is the caller graph for this function:

8.12.3.9 void readlines_ctx_free (readlines_ctx *ctx*)

Definition at line 331 of file devnode.c.

References _readlines_ctx::buffer.

Referenced by devmain().

```
331                                         {
332     free(ctx->buffer);
333     free(ctx);
334 }
```

Here is the caller graph for this function:

8.12.3.10 void readlines_ctx_init (readlines_ctx * *ctx*)

Definition at line 324 of file devnode.c.

Referenced by devmain().

```
324                                          {
325     // Allocate buffers to store data
326     *ctx = calloc(1, sizeof(struct _readlines_ctx));
327     int buffersize = (*ctx)->buffersize = 4095;
328     (*ctx)->buffer = malloc(buffersize + 1);
329 }
```

Here is the caller graph for this function:

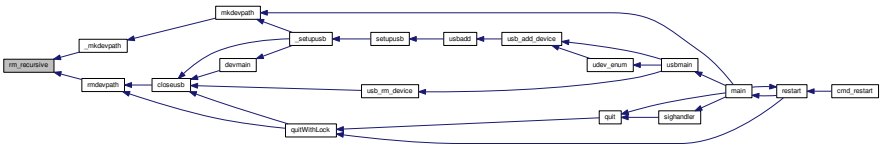
8.12.3.11 int rm_recursive (const char * *path*)

Definition at line 19 of file devnode.c.

Referenced by _mkdevpath(), and rmdevpath().

```
19                                  {
20     DIR* dir = opendir(path);
21     if(!dir)
22         return remove(path);
23     struct dirent* file;
24     while((file = readdir(dir)))
25     {
26         if(!strcmp(file->d_name, ".") || !strcmp(file->d_name, ".."))
27             continue;
28         char path2[FILENAME_MAX];
29         snprintf(path2, FILENAME_MAX, "%s/%s", path, file->d_name);
30         int stat = rm_recursive(path2);
31         if(stat != 0)
32             return stat;
33     }
34     closedir(dir);
35     return remove(path);
36 }
```

Here is the caller graph for this function:



8.12.3.12 int rmdevpath (usbdevice * *kb*)

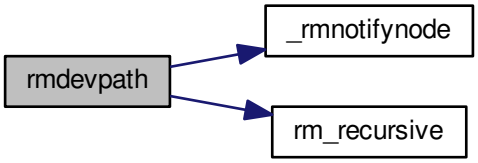
Definition at line 258 of file devnode.c.

References _rmnotifynode(), ckb_info, ckb_warn, devpath, euid_guard_start, euid_guard_stop, INDEX_OF, usbdevice::infifo, keyboard, OUTFIFO_MAX, and rm_recursive().

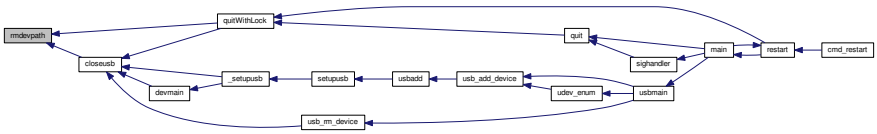
Referenced by closeusb(), and quitWithLock().

```
258                                  {
259     euid_guard_start;
260     int index = INDEX_OF(kb, keyboard);
261     if(kb->infifo != 0){
262 #ifdef OS_LINUX
263         write(kb->infifo - 1, "\n", 1); // hack to prevent the FIFO thread from perma-blocking
264 #endif
265         close(kb->infifo - 1);
266         kb->infifo = 0;
267     }
268     for(int i = 0; i < OUTFIFO_MAX; i++)
269         _rmnotifynode(kb, i);
270     char path[strlen(devpath) + 2];
271     snprintf(path, sizeof(path), "%s%d", devpath, index);
272     if(rm_recursive(path) != 0 && errno != ENOENT){
273         ckb_warn("Unable to delete %s: %s\n", path, strerror(errno));
274         euid_guard_stop;
275         return -1;
276     }
277     ckb_info("Removed device path %s\n", path);
278     euid_guard_stop;
279     return 0;
280 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



8.12.3.13 int rmnotifynode (usbdevice * *kb*, int *notify*)

Definition at line 112 of file devnode.c.

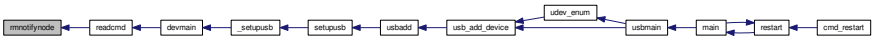
References _rmnotifynode(), euid_guard_start, and euid_guard_stop.

Referenced by readcmd().

```
112                                             {
113     euid_guard_start;
114     int res = _rmnotifynode(kb, notify);
115     euid_guard_stop;
116     return res;
117 }
```

Here is the call graph for this function:

Here is the caller graph for this function:



8.12.3.14 void updateconnected ( )

Definition at line 64 of file devnode.c.

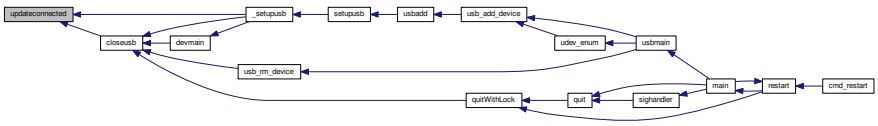
References _updateconnected(), euid_guard_start, and euid_guard_stop.

Referenced by _setupusb(), and closeusb().

```
64                          {
65     euid_guard_start;
66     _updateconnected();
67     euid_guard_stop;
68 }
```

Here is the call graph for this function:

Here is the caller graph for this function:



8.12.4 Variable Documentation

8.12.4.1 const char* const devpath = "/dev/input/ckb"

Definition at line 11 of file devnode.c.

Referenced by _mkdevpath(), _mknotifynode(), _rmnotifynode(), _setupusb(), _updateconnected(), closeusb(), main(), mkfwnode(), os_inputmain(), os_setupusb(), and rmdevpath().

8.12.4.2 long gid = -1

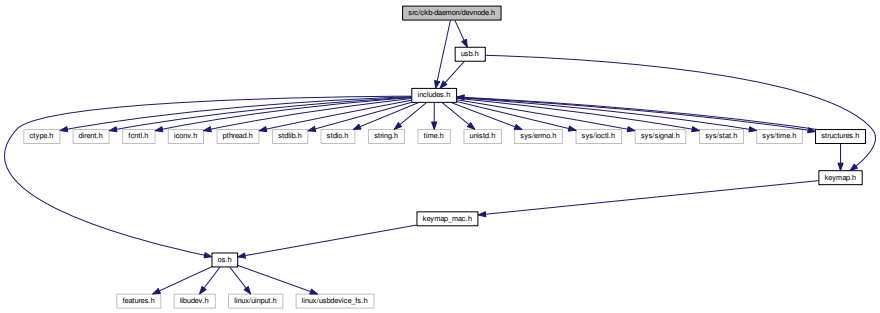
Definition at line 16 of file devnode.c.

Referenced by _mkdevpath(), _mknotifynode(), _updateconnected(), main(), and mkfwnode().

8.13 src/ckb-daemon/devnode.h File Reference

```
#include "includes.h"
#include "usb.h"
```

Include dependency graph for devnode.h:



This graph shows which files directly or indirectly include this file:

Macros

- #define S_READDIR (S_IRWXU | S_IRGRP | S_IROTH | S_IXGRP | S_IXOTH)
- #define S_READ (S_IRUSR | S_IRGRP | S_IROTH | S_IWUSR)
- #define S_READWRITE (S_IRUSR | S_IRGRP | S_IROTH | S_IWUSR | S_IWGRP | S_IWOTH)
- #define S_CUSTOM (S_IRUSR | S_IWUSR | S_IRGRP | S_IWGRP)
- #define S_CUSTOM_R (S_IRUSR | S_IWUSR | S_IRGRP)

Typedefs

- typedef struct _readlines_ctx * readlines_ctx

Functions

- void updateconnected ()
- int mkdevpath (usbdevice *kb)
- int rmdevpath (usbdevice *kb)
- int mknotifynode (usbdevice *kb, int notify)
- int rmnotifynode (usbdevice *kb, int notify)
- int mkfwnode (usbdevice *kb)
- void readlines_ctx_init (readlines_ctx *ctx)
- void readlines_ctx_free (readlines_ctx ctx)
- unsigned readlines (int fd, readlines_ctx ctx, const char **input)

Variables

- const char *const devpath
- long gid

8.13.1 Macro Definition Documentation

8.13.1.1 #define S_CUSTOM (S_IRUSR | S_IWUSR | S_IRGRP | S_IWGRP)

Definition at line 17 of file devnode.h.

Referenced by _mkdevpath().

8.13.1.2 #define S_CUSTOM_R (S_IRUSR | S_IWUSR | S_IRGRP)

Definition at line 18 of file devnode.h.

8.13.1.3 #define S_READ (S_IRUSR | S_IRGRP | S_IROTH | S_IWUSR)

Definition at line 15 of file devnode.h.

Referenced by _mkdevpath().

8.13.1.4 #define S_READDIR (S_IRWXU | S_IRGRP | S_IROTH | S_IXGRP | S_IXOTH)

Definition at line 14 of file devnode.h.

Referenced by _mkdevpath().

8.13.1.5 #define S_READWRITE (S_IRUSR | S_IRGRP | S_IROTH | S_IWUSR | S_IWGRP | S_IWOTH)

Definition at line 16 of file devnode.h.

Referenced by _mkdevpath().

8.13.2 Typedef Documentation

8.13.2.1 typedef struct _readlines_ctx* readlines_ctx

Definition at line 36 of file devnode.h.

8.13.3 Function Documentation

8.13.3.1 int mkdevpath (usbdevice * kb)

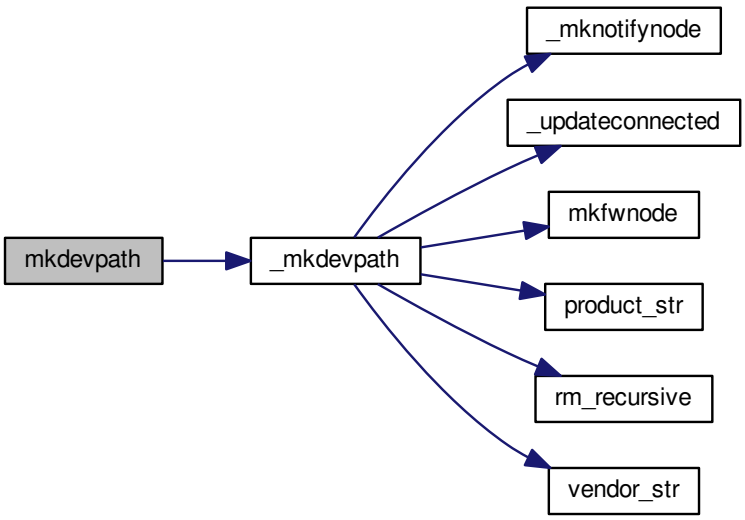
Definition at line 251 of file devnode.c.

References _mkdevpath(), euid_guard_start, and euid_guard_stop.

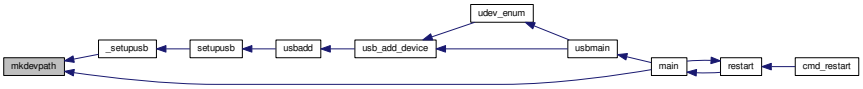
Referenced by _setupusb(), and main().

```
251                             {
252     euid_guard_start;
253     int res = _mkdevpath(kb);
254     euid_guard_stop;
255     return res;
256 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



8.13.3.2 int mkfwnode (usbdevice * *kb*)

Definition at line 282 of file devnode.c.

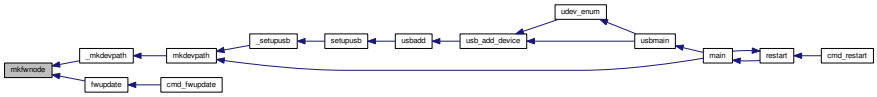
References ckb_warn, devpath, usbdevice::fwversion, gid, INDEX_OF, keyboard, usbdevice::pollrate, and S_GID_-READ.

Referenced by _mkdevpath(), and fwupdate().

```
282                          {
283     int index = INDEX_OF(kb, keyboard);
284     char fwpath[strlen(devpath) + 12];
285     snprintf(fwpath, sizeof(fwpath), "%s%d/fwversion", devpath, index);
286     FILE* fwfile = fopen(fwpath, "w");
287     if(fwfile){
288         fprintf(fwfile, "%04x", kb->fwversion);
289         fputc('\n', fwfile);
290         fclose(fwfile);
291         chmod(fwpath, S_GID_READ);
292         if(gid >= 0)
293             chown(fwpath, 0, gid);
294     } else {
295         ckb_warn("Unable to create %s: %s\n", fwpath, strerror(errno));
296         remove(fwpath);
297         return -1;
298     }
299     char ppath[strlen(devpath) + 11];
300     snprintf(ppath, sizeof(ppath), "%s%d/pollrate", devpath, index);
301     FILE* pfile = fopen(ppath, "w");
```

```
302     if(pfile){
303         fprintf(pfile, "%d ms", kb->pollrate);
304         fputc('\n', pfile);
305         fclose(pfile);
306         chmod(ppath, S_GID_READ);
307         if(gid >= 0)
308             chown(ppath, 0, gid);
309     } else {
310         ckb_warn("Unable to create %s: %s\n", fwpath, strerror(errno));
311         remove(ppath);
312         return -2;
313     }
314     return 0;
315 }
```

Here is the caller graph for this function:



8.13.3.3 int mknotifynode (usbdevice * *kb*, int *notify*)

Definition at line 91 of file devnode.c.

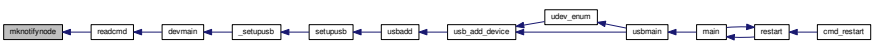
References _mknotifynode(), euid_guard_start, and euid_guard_stop.

Referenced by readcmd().

```
91                                          {
92     euid_guard_start;
93     int res = _mknotifynode(kb, notify);
94     euid_guard_stop;
95     return res;
96 }
```

Here is the call graph for this function:

Here is the caller graph for this function:



8.13.3.4 unsigned readlines (int *fd*, readlines_ctx *ctx*, const char ** *input*)

Definition at line 336 of file devnode.c.

References _readlines_ctx::buffer, _readlines_ctx::buffersize, ckb_warn, _readlines_ctx::leftover, _readlines_ctx::leftoverlen, and MAX_BUFFER.

Referenced by devmain().

```
336                                                                 {
337     // Move any data left over from a previous read to the start of the buffer
338     char* buffer = ctx->buffer;
339     int buffersize = ctx->buffersize;
340     int leftover = ctx->leftover, leftoverlen = ctx->leftoverlen;
341     memcpy(buffer, buffer + leftover, leftoverlen);
342     // Read data from the file
343     ssize_t length = read(fd, buffer + leftoverlen, buffersize - leftoverlen);
344     length = (length < 0 ? 0 : length) + leftoverlen;
345     leftover = ctx->leftover = leftoverlen = ctx->leftoverlen = 0;
346     if(length <= 0){
347         *input = 0;
348         return 0;
349     }
350     // Continue buffering until all available input is read or there's no room left
351     while(length == buffersize){
352         if(buffersize == MAX_BUFFER)
353             break;
354         int oldsize = buffersize;
355         buffersize += 4096;
356         ctx->buffersize = buffersize;
357         buffer = ctx->buffer = realloc(buffer, buffersize + 1);
358         ssize_t length2 = read(fd, buffer + oldsize, buffersize - oldsize);
359         if(length2 <= 0)
360             break;
361         length += length2;
362     }
363     buffer[length] = 0;
364     // Input should be issued one line at a time and should end with a newline.
365     char* lastline = memrchr(buffer, '\n', length);
366     if(lastline == buffer + length - 1){
367         // If the buffer ends in a newline, process the whole string
368         *input = buffer;
369         return length;
370     } else if(lastline){
371         // Otherwise, chop off the last line but process everything else
372         *lastline = 0;
373         leftover = ctx->leftover = lastline + 1 - buffer;
374         leftoverlen = ctx->leftoverlen = length - leftover;
375         *input = buffer;
376         return leftover - 1;
377     } else {
378         // If a newline wasn't found at all, process the whole buffer next time
379         *input = 0;
380         if(length == MAX_BUFFER){
381             // Unless the buffer is completely full, in which case discard it
382             ckb_warn("Too much input (1MB). Dropping.\n");
383             return 0;
384         }
385         leftoverlen = ctx->leftoverlen = length;
386         return 0;
387     }
388 }
```

Here is the caller graph for this function:

8.13.3.5 void readlines_ctx_free (readlines_ctx *ctx*)

Definition at line 331 of file devnode.c.

References _readlines_ctx::buffer.

Referenced by devmain().

```
331                                             {
```

```
332     free(ctx->buffer);
333     free(ctx);
334 }
```

Here is the caller graph for this function:

8.13.3.6 void readlines_ctx_init (readlines_ctx * ctx)

Definition at line 324 of file devnode.c.

Referenced by devmain().

```
324                                         {
325     // Allocate buffers to store data
326     *ctx = calloc(1, sizeof(struct _readlines_ctx));
327     int buffersize = (*ctx)->buffersize = 4095;
328     (*ctx)->buffer = malloc(buffersize + 1);
329 }
```

Here is the caller graph for this function:

8.13.3.7 int rmdevpath (usbdevice * kb)

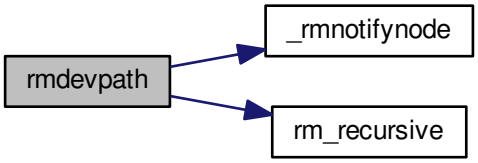
Definition at line 258 of file devnode.c.

References _rmnotifynode(), ckb_info, ckb_warn, devpath, euid_guard_start, euid_guard_stop, INDEX_OF, usbdevice::infifo, keyboard, OUTFIFO_MAX, and rm_recursive().

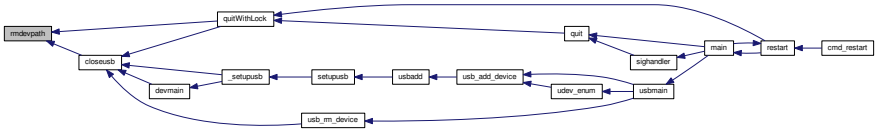
Referenced by closeusb(), and quitWithLock().

```
258                             {
259     euid_guard_start;
260     int index = INDEX_OF(kb, keyboard);
261     if(kb->infifo != 0){
262 #ifdef OS_LINUX
263         write(kb->infifo - 1, "\n", 1); // hack to prevent the FIFO thread from perma-blocking
264 #endif
265         close(kb->infifo - 1);
266         kb->infifo = 0;
267     }
268     for(int i = 0; i < OUTFIFO_MAX; i++)
269         _rmnotifynode(kb, i);
270     char path[strlen(devpath) + 2];
271     snprintf(path, sizeof(path), "%s%d", devpath, index);
272     if(rm_recursive(path) != 0 && errno != ENOENT){
273         ckb_warn("Unable to delete %s: %s\n", path, strerror(errno));
274         euid_guard_stop;
275         return -1;
276     }
277     ckb_info("Removed device path %s\n", path);
278     euid_guard_stop;
279     return 0;
280 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



8.13.3.8 int rmnotifynode (usbdevice * *kb*, int *notify*)

Definition at line 112 of file devnode.c.

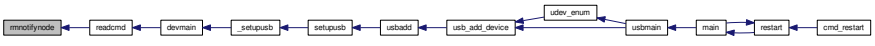
References _rmnotifynode(), euid_guard_start, and euid_guard_stop.

Referenced by readcmd().

```
112                                             {
113     euid_guard_start;
114     int res = _rmnotifynode(kb, notify);
115     euid_guard_stop;
116     return res;
117 }
```

Here is the call graph for this function:

Here is the caller graph for this function:



8.13.3.9 void updateconnected ( )

Definition at line 64 of file devnode.c.

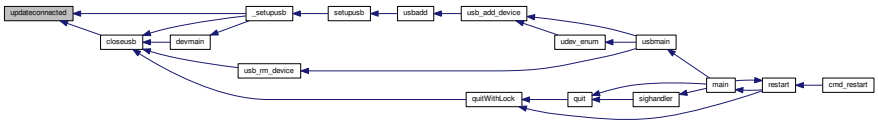
References _updateconnected(), euid_guard_start, and euid_guard_stop.

Referenced by _setupusb(), and closeusb().

```
64                          {
65     euid_guard_start;
66     _updateconnected();
67     euid_guard_stop;
68 }
```

Here is the call graph for this function:

Here is the caller graph for this function:



8.13.4 Variable Documentation

8.13.4.1 const char* const devpath

Definition at line 8 of file devnode.h.

8.13.4.2 long gid

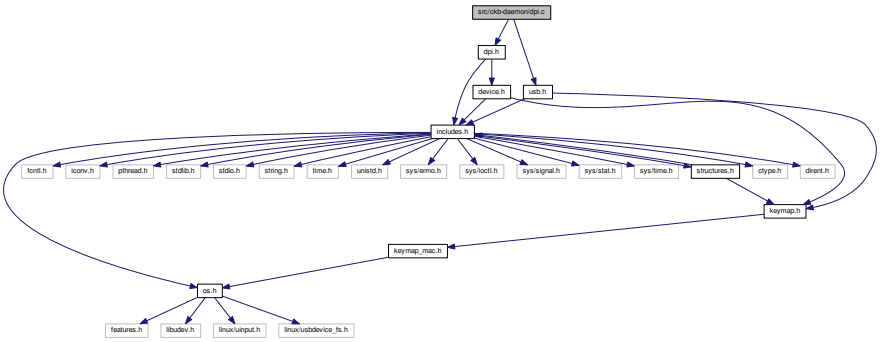
Definition at line 16 of file devnode.c.

Referenced by _mkdevpath(), _mknotifynode(), _updateconnected(), main(), and mkfwnode().

8.14 src/ckb-daemon/dpi.c File Reference

```
#include "dpi.h"
#include "usb.h"
```

Include dependency graph for dpi.c:



Functions

- void cmd_dpi (usbdevice ∗kb, usbmode ∗mode, int dummy, const char ∗stages, const char ∗values)
- void cmd_dpisel (usbdevice ∗kb, usbmode ∗mode, int dummy1, int dummy2, const char ∗stage)
- void cmd_lift (usbdevice ∗kb, usbmode ∗mode, int dummy1, int dummy2, const char ∗height)
- void cmd_snap (usbdevice ∗kb, usbmode ∗mode, int dummy1, int dummy2, const char ∗enable)
- char ∗ printdpi (const dpiset ∗dpi, const usbdevice ∗kb)
- int updatedpi (usbdevice ∗kb, int force)
- int savedpi (usbdevice ∗kb, dpiset ∗dpi, lighting ∗light)
- int loaddpi (usbdevice ∗kb, dpiset ∗dpi, lighting ∗light)

8.14.1 Function Documentation

8.14.1.1 void cmd_dpi (usbdevice ∗ *kb,* usbmode ∗ *mode,* int *dummy,* const char ∗ *stages,* const char ∗ *values*)

Definition at line 4 of file dpi.c.

References usbmode::dpi, DPI_COUNT, dpiset::enabled, dpiset::x, and dpiset::y.

```
4                                                                                                                {
5       int disable = 0;
6       ushort x, y;
7       // Try to scan X,Y values
8       if(sscanf(values, "%hu,%hu", &x, &y) != 2){
9           // If that doesn't work, scan single number
10          if(sscanf(values, "%hu", &x) == 1)
11              y = x;
12          else if(!strncmp(values, "off", 3))
13              // If the right side says "off", disable the level(s)
14              disable = 1;
15          else
16              // Otherwise, quit
17              return;
18      }
19      if((x == 0 || y == 0) && !disable)
20          return;
21      // Scan the left side for stage numbers (comma-separated)
22      int left = strlen(stages);
23      int position = 0, field = 0;
24      char stagename[3];
25      while(position < left && sscanf(stages + position, "%2[^,]%n", stagename, &field) == 1){
26          uchar stagenum;
27          if(sscanf(stagename, "%hhu", &stagenum) && stagenum < DPI_COUNT){
28              // Set DPI for this stage
29              if(disable){
30                  mode->dpi.enabled &= ~(1 << stagenum);
31                  mode->dpi.x[stagenum] = 0;
32                  mode->dpi.y[stagenum] = 0;
33              } else {
34                  mode->dpi.enabled |= 1 << stagenum;
35                  mode->dpi.x[stagenum] = x;
```

```
36                 mode->dpi.y[stagenum] = y;
37             }
38         }
39         if(stages[position += field] == ',')
40             position++;
41     }
42 }
```

8.14.1.2 void cmd_dpisel (usbdevice * *kb*, usbmode * *mode*, int *dummy1*, int *dummy2*, const char * *stage*)

Definition at line 44 of file dpi.c.

References dpiset::current, usbmode::dpi, and DPI_COUNT.

```
44                                                                                  {
45     uchar stagenum;
46     if(sscanf(stage, "%hhu", &stagenum) != 1)
47         return;
48     if(stagenum > DPI_COUNT)
49         return;
50     mode->dpi.current = stagenum;
51 }
```

8.14.1.3 void cmd_lift (usbdevice * *kb*, usbmode * *mode*, int *dummy1*, int *dummy2*, const char * *height*)

Definition at line 53 of file dpi.c.

References usbmode::dpi, dpiset::lift, LIFT_MAX, and LIFT_MIN.

```
53                                                                                  {
54     uchar heightnum;
55     if(sscanf(height, "%hhu", &heightnum) != 1)
56         return;
57     if(heightnum > LIFT_MAX || heightnum < LIFT_MIN)
58         return;
59     mode->dpi.lift = heightnum;
60 }
```

8.14.1.4 void cmd_snap (usbdevice * *kb*, usbmode * *mode*, int *dummy1*, int *dummy2*, const char * *enable*)

Definition at line 62 of file dpi.c.

References usbmode::dpi, and dpiset::snap.

```
62                                                                                  {
63     if(!strcmp(enable, "on"))
64         mode->dpi.snap = 1;
65     if(!strcmp(enable, "off"))
66         mode->dpi.snap = 0;
67 }
```

8.14.1.5 int loaddpi (usbdevice * *kb*, dpiset * *dpi*, lighting * *light*)

Definition at line 152 of file dpi.c.

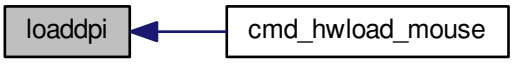
References lighting::b, ckb_err, dpiset::current, DPI_COUNT, dpiset::enabled, lighting::g, LED_MOUSE, dpiset::lift, LIFT_MAX, LIFT_MIN, MSG_SIZE, N_MOUSE_ZONES, lighting::r, dpiset::snap, usbrecv, dpiset::x, and dpiset::y.

Referenced by cmd_hwload_mouse().

```
152                                                 {
153     // Ask for settings
154     uchar data_pkt[4][MSG_SIZE] = {
155         { 0x0e, 0x13, 0x05, 1, },
156         { 0x0e, 0x13, 0x02, 1, },
```

```
157         { 0x0e, 0x13, 0x03, 1, },
158         { 0x0e, 0x13, 0x04, 1, }
159     };
160     uchar in_pkt[4][MSG_SIZE];
161     for(int i = 0; i < 4; i++){
162         if(!usbrecv(kb, data_pkt[i], in_pkt[i]))
163             return -2;
164         if(memcmp(in_pkt[i], data_pkt[i], 4)){
165             ckb_err("Bad input header\n");
166             return -3;
167         }
168     }
169     // Copy data from device
170     dpi->enabled = in_pkt[0][4];
171     dpi->enabled &= (1 << DPI_COUNT) - 1;
172     dpi->current = in_pkt[1][4];
173     if(dpi->current >= DPI_COUNT)
174         dpi->current = 0;
175     dpi->lift = in_pkt[2][4];
176     if(dpi->lift < LIFT_MIN || dpi->lift > LIFT_MAX)
177         dpi->lift = LIFT_MIN;
178     dpi->snap = !!in_pkt[3][4];
179
180     // Get X/Y DPIs
181     for(int i = 0; i < DPI_COUNT; i++){
182         uchar data_pkt[MSG_SIZE] = { 0x0e, 0x13, 0xd0, 1 };
183         uchar in_pkt[MSG_SIZE];
184         data_pkt[2] |= i;
185         if(!usbrecv(kb, data_pkt, in_pkt))
186             return -2;
187         if(memcmp(in_pkt, data_pkt, 4)){
188             ckb_err("Bad input header\n");
189             return -3;
190         }
191         // Copy to profile
192         dpi->x[i] = *(ushort*)(in_pkt + 5);
193         dpi->y[i] = *(ushort*)(in_pkt + 7);
194         light->r[LED_MOUSE + N_MOUSE_ZONES + i] = in_pkt[9];
195         light->g[LED_MOUSE + N_MOUSE_ZONES + i] = in_pkt[10];
196         light->b[LED_MOUSE + N_MOUSE_ZONES + i] = in_pkt[11];
197     }
198     // Finished. Set SW DPI light to the current hardware level
199     light->r[LED_MOUSE + 2] = light->r[LED_MOUSE +
    N_MOUSE_ZONES + dpi->current];
200     light->g[LED_MOUSE + 2] = light->g[LED_MOUSE +
    N_MOUSE_ZONES + dpi->current];
201     light->b[LED_MOUSE + 2] = light->b[LED_MOUSE +
    N_MOUSE_ZONES + dpi->current];
202     return 0;
203 }
```

Here is the caller graph for this function:



8.14.1.6 char* printdpi (const dpiset * *dpi,* const usbdevice * *kb*)

Definition at line 69 of file dpi.c.

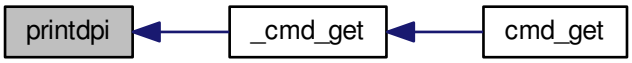
References _readlines_ctx::buffer, DPI_COUNT, dpiset::enabled, dpiset::x, and dpiset::y.

Referenced by _cmd_get().

```
69                                                  {
70     // Print all DPI settings
71     const int BUFFER_LEN = 100;
```

```
72     char* buffer = malloc(BUFFER_LEN);
73     int length = 0;
74     for(int i = 0; i < DPI_COUNT; i++){
75         // Print the stage number
76         int newlen = 0;
77         snprintf(buffer + length, BUFFER_LEN - length, length == 0 ? "%d%n" : " %d%n", i, &newlen);
78         length += newlen;
79         // Print the DPI settings
80         if(!(dpi->enabled & (1 << i)))
81             snprintf(buffer + length, BUFFER_LEN - length, ":off%n", &newlen);
82         else
83             snprintf(buffer + length, BUFFER_LEN - length, ":%u,%u%n", dpi->x[i], dpi->
     y[i], &newlen);
84         length += newlen;
85     }
86     return buffer;
87 }
```

Here is the caller graph for this function:



8.14.1.7 int savedpi (usbdevice * *kb*, dpiset * *dpi*, lighting * *light*)

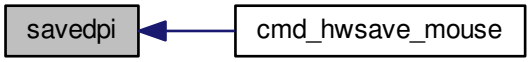
Definition at line 124 of file dpi.c.

References lighting::b, dpiset::current, DPI_COUNT, dpiset::enabled, lighting::g, LED_MOUSE, dpiset::lift, MSG_-SIZE, N_MOUSE_ZONES, lighting::r, dpiset::snap, usbsend, dpiset::x, and dpiset::y.

Referenced by cmd_hwsave_mouse().

```
124                                                 {
125     // Send X/Y DPIs
126     for(int i = 0; i < DPI_COUNT; i++){
127         uchar data_pkt[MSG_SIZE] = { 0x07, 0x13, 0xd0, 1 };
128         data_pkt[2] |= i;
129         *(ushort*)(data_pkt + 5) = dpi->x[i];
130         *(ushort*)(data_pkt + 7) = dpi->y[i];
131         // Save the RGB value for this setting too
132         data_pkt[9] = light->r[LED_MOUSE + N_MOUSE_ZONES + i];
133         data_pkt[10] = light->g[LED_MOUSE + N_MOUSE_ZONES + i];
134         data_pkt[11] = light->b[LED_MOUSE + N_MOUSE_ZONES + i];
135         if(!usbsend(kb, data_pkt, 1))
136             return -1;
137     }
138
139     // Send settings
140     uchar data_pkt[4][MSG_SIZE] = {
141         { 0x07, 0x13, 0x05, 1, dpi->enabled },
142         { 0x07, 0x13, 0x02, 1, dpi->current },
143         { 0x07, 0x13, 0x03, 1, dpi->lift },
144         { 0x07, 0x13, 0x04, 1, dpi->snap, 0x05 }
145     };
146     if(!usbsend(kb, data_pkt[0], 4))
147         return -2;
148     // Finished
149     return 0;
150 }
```

Here is the caller graph for this function:



8.14.1.8 int updatedpi (usbdevice * *kb*, int *force*)

Definition at line 89 of file dpi.c.

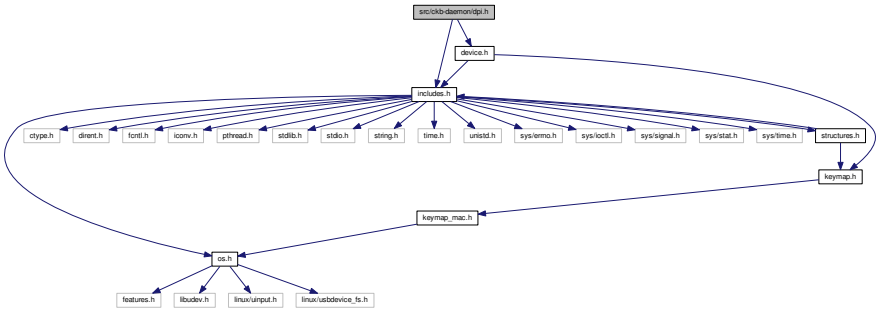
References usbdevice::active, dpiset::current, usbprofile::currentmode, usbmode::dpi, DPI_COUNT, dpiset::enabled, dpiset::forceupdate, usbprofile::lastdpi, dpiset::lift, MSG_SIZE, usbdevice::profile, dpiset::snap, usbsend, dpiset::x, and dpiset::y.

```
89                                      {
90     if(!kb->active)
91         return 0;
92     dpiset* lastdpi = &kb->profile->lastdpi;
93     dpiset* newdpi = &kb->profile->currentmode->dpi;
94     // Don't do anything if the settings haven't changed
95     if(!force && !lastdpi->forceupdate && !newdpi->forceupdate
96             && !memcmp(lastdpi, newdpi, sizeof(dpiset)))
97         return 0;
98     lastdpi->forceupdate = newdpi->forceupdate = 0;
99
100     // Send X/Y DPIs
101     for(int i = 0; i < DPI_COUNT; i++){
102         uchar data_pkt[MSG_SIZE] = { 0x07, 0x13, 0xd0, 0 };
103         data_pkt[2] |= i;
104         *(ushort*)(data_pkt + 5) = newdpi->x[i];
105         *(ushort*)(data_pkt + 7) = newdpi->y[i];
106         if(!usbsend(kb, data_pkt, 1))
107             return -1;
108     }
109
110     // Send settings
111     uchar data_pkt[4][MSG_SIZE] = {
112         { 0x07, 0x13, 0x05, 0, newdpi->enabled },
113         { 0x07, 0x13, 0x02, 0, newdpi->current },
114         { 0x07, 0x13, 0x03, 0, newdpi->lift },
115         { 0x07, 0x13, 0x04, 0, newdpi->snap, 0x05 }
116     };
117     if(!usbsend(kb, data_pkt[0], 4))
118         return -2;
119     // Finished
120     memcpy(lastdpi, newdpi, sizeof(dpiset));
121     return 0;
122 }
```

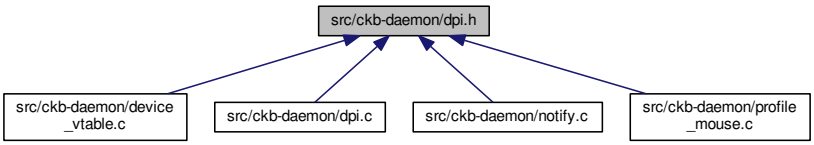
8.15 src/ckb-daemon/dpi.h File Reference

```
#include "includes.h"
#include "device.h"
```

Include dependency graph for dpi.h:



This graph shows which files directly or indirectly include this file:



Functions

- int updatedpi (usbdevice ∗kb, int force)
- int savedpi (usbdevice ∗kb, dpiset ∗dpi, lighting ∗light)
- int loaddpi (usbdevice ∗kb, dpiset ∗dpi, lighting ∗light)
- char ∗ printdpi (const dpiset ∗dpi, const usbdevice ∗kb)
- void cmd_dpi (usbdevice ∗kb, usbmode ∗mode, int dummy, const char ∗stages, const char ∗values)
- void cmd_dpisel (usbdevice ∗kb, usbmode ∗mode, int dummy1, int dummy2, const char ∗stage)
- void cmd_lift (usbdevice ∗kb, usbmode ∗mode, int dummy1, int dummy2, const char ∗height)
- void cmd_snap (usbdevice ∗kb, usbmode ∗mode, int dummy1, int dummy2, const char ∗enable)

8.15.1 Function Documentation

8.15.1.1 void cmd_dpi (usbdevice ∗ *kb,* usbmode ∗ *mode,* int *dummy,* const char ∗ *stages,* const char ∗ *values*)

Definition at line 4 of file dpi.c.

References usbmode::dpi, DPI_COUNT, dpiset::enabled, dpiset::x, and dpiset::y.

```
4                                                                                                   {
5       int disable = 0;
6       ushort x, y;
7       // Try to scan X,Y values
8       if(sscanf(values, "%hu,%hu", &x, &y) != 2){
9           // If that doesn't work, scan single number
10          if(sscanf(values, "%hu", &x) == 1)
11              y = x;
12          else if(!strncmp(values, "off", 3))
13              // If the right side says "off", disable the level(s)
14              disable = 1;
15          else
16              // Otherwise, quit
17              return;
```

```
18      }
19      if((x == 0 || y == 0) && !disable)
20          return;
21      // Scan the left side for stage numbers (comma-separated)
22      int left = strlen(stages);
23      int position = 0, field = 0;
24      char stagename[3];
25      while(position < left && sscanf(stages + position, "%2[^,]%n", stagename, &field) == 1){
26          uchar stagenum;
27          if(sscanf(stagename, "%hhu", &stagenum) && stagenum < DPI_COUNT){
28              // Set DPI for this stage
29              if(disable){
30                  mode->dpi.enabled &= ~(1 << stagenum);
31                  mode->dpi.x[stagenum] = 0;
32                  mode->dpi.y[stagenum] = 0;
33              } else {
34                  mode->dpi.enabled |= 1 << stagenum;
35                  mode->dpi.x[stagenum] = x;
36                  mode->dpi.y[stagenum] = y;
37              }
38          }
39          if(stages[position += field] == ',')
40              position++;
41      }
42  }
```

8.15.1.2 void cmd_dpisel (usbdevice * *kb*, usbmode * *mode*, int *dummy1*, int *dummy2*, const char * *stage*)

Definition at line 44 of file dpi.c.

References dpiset::current, usbmode::dpi, and DPI_COUNT.

```
44                                                                                  {
45      uchar stagenum;
46      if(sscanf(stage, "%hhu", &stagenum) != 1)
47          return;
48      if(stagenum > DPI_COUNT)
49          return;
50      mode->dpi.current = stagenum;
51  }
```

8.15.1.3 void cmd_lift (usbdevice * *kb*, usbmode * *mode*, int *dummy1*, int *dummy2*, const char * *height*)

Definition at line 53 of file dpi.c.

References usbmode::dpi, dpiset::lift, LIFT_MAX, and LIFT_MIN.

```
53                                                                                  {
54      uchar heightnum;
55      if(sscanf(height, "%hhu", &heightnum) != 1)
56          return;
57      if(heightnum > LIFT_MAX || heightnum < LIFT_MIN)
58          return;
59      mode->dpi.lift = heightnum;
60  }
```

8.15.1.4 void cmd_snap (usbdevice * *kb*, usbmode * *mode*, int *dummy1*, int *dummy2*, const char * *enable*)

Definition at line 62 of file dpi.c.

References usbmode::dpi, and dpiset::snap.

```
62                                                                                  {
63      if(!strcmp(enable, "on"))
64          mode->dpi.snap = 1;
65      if(!strcmp(enable, "off"))
66          mode->dpi.snap = 0;
67  }
```

8.15.1.5 int loaddpi (usbdevice * *kb,* dpiset * *dpi,* lighting * *light*)

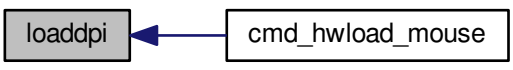
Definition at line 152 of file dpi.c.

References lighting::b, ckb_err, dpiset::current, DPI_COUNT, dpiset::enabled, lighting::g, LED_MOUSE, dpiset::lift, LIFT_MAX, LIFT_MIN, MSG_SIZE, N_MOUSE_ZONES, lighting::r, dpiset::snap, usbrecv, dpiset::x, and dpiset::y.

Referenced by cmd_hwload_mouse().

```
152                                                     {
153     // Ask for settings
154     uchar data_pkt[4][MSG_SIZE] = {
155         { 0x0e, 0x13, 0x05, 1, },
156         { 0x0e, 0x13, 0x02, 1, },
157         { 0x0e, 0x13, 0x03, 1, },
158         { 0x0e, 0x13, 0x04, 1, }
159     };
160     uchar in_pkt[4][MSG_SIZE];
161     for(int i = 0; i < 4; i++){
162         if(!usbrecv(kb, data_pkt[i], in_pkt[i]))
163             return -2;
164         if(memcmp(in_pkt[i], data_pkt[i], 4)){
165             ckb_err("Bad input header\n");
166             return -3;
167         }
168     }
169     // Copy data from device
170     dpi->enabled = in_pkt[0][4];
171     dpi->enabled &= (1 << DPI_COUNT) - 1;
172     dpi->current = in_pkt[1][4];
173     if(dpi->current >= DPI_COUNT)
174         dpi->current = 0;
175     dpi->lift = in_pkt[2][4];
176     if(dpi->lift < LIFT_MIN || dpi->lift > LIFT_MAX)
177         dpi->lift = LIFT_MIN;
178     dpi->snap = !!in_pkt[3][4];
179
180     // Get X/Y DPIs
181     for(int i = 0; i < DPI_COUNT; i++){
182         uchar data_pkt[MSG_SIZE] = { 0x0e, 0x13, 0xd0, 1 };
183         uchar in_pkt[MSG_SIZE];
184         data_pkt[2] |= i;
185         if(!usbrecv(kb, data_pkt, in_pkt))
186             return -2;
187         if(memcmp(in_pkt, data_pkt, 4)){
188             ckb_err("Bad input header\n");
189             return -3;
190         }
191         // Copy to profile
192         dpi->x[i] = *(ushort*)(in_pkt + 5);
193         dpi->y[i] = *(ushort*)(in_pkt + 7);
194         light->r[LED_MOUSE + N_MOUSE_ZONES + i] = in_pkt[9];
195         light->g[LED_MOUSE + N_MOUSE_ZONES + i] = in_pkt[10];
196         light->b[LED_MOUSE + N_MOUSE_ZONES + i] = in_pkt[11];
197     }
198     // Finished. Set SW DPI light to the current hardware level
199     light->r[LED_MOUSE + 2] = light->r[LED_MOUSE +
    N_MOUSE_ZONES + dpi->current];
200     light->g[LED_MOUSE + 2] = light->g[LED_MOUSE +
    N_MOUSE_ZONES + dpi->current];
201     light->b[LED_MOUSE + 2] = light->b[LED_MOUSE +
    N_MOUSE_ZONES + dpi->current];
202     return 0;
203 }
```

Here is the caller graph for this function:



8.15.1.6 char* printdpi (const dpiset * *dpi,* const usbdevice * *kb*)

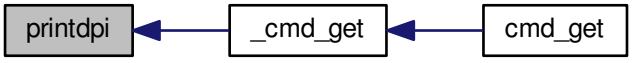
Definition at line 69 of file dpi.c.

References _readlines_ctx::buffer, DPI_COUNT, dpiset::enabled, dpiset::x, and dpiset::y.

Referenced by _cmd_get().

```
69                                                      {
70     // Print all DPI settings
71     const int BUFFER_LEN = 100;
72     char* buffer = malloc(BUFFER_LEN);
73     int length = 0;
74     for(int i = 0; i < DPI_COUNT; i++){
75         // Print the stage number
76         int newlen = 0;
77         snprintf(buffer + length, BUFFER_LEN - length, length == 0 ? "%d%n" : " %d%n", i, &newlen);
78         length += newlen;
79         // Print the DPI settings
80         if(!(dpi->enabled & (1 << i)))
81             snprintf(buffer + length, BUFFER_LEN - length, ":off%n", &newlen);
82         else
83             snprintf(buffer + length, BUFFER_LEN - length, ":%u,%u%n", dpi->x[i], dpi->
       y[i], &newlen);
84         length += newlen;
85     }
86     return buffer;
87 }
```

Here is the caller graph for this function:



8.15.1.7 int savedpi (usbdevice * *kb,* dpiset * *dpi,* lighting * *light*)

Definition at line 124 of file dpi.c.

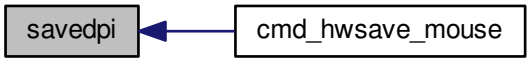
References lighting::b, dpiset::current, DPI_COUNT, dpiset::enabled, lighting::g, LED_MOUSE, dpiset::lift, MSG_-SIZE, N_MOUSE_ZONES, lighting::r, dpiset::snap, usbsend, dpiset::x, and dpiset::y.

Referenced by cmd_hwsave_mouse().

```
124                                                       {
125     // Send X/Y DPIs
126     for(int i = 0; i < DPI_COUNT; i++){
127         uchar data_pkt[MSG_SIZE] = { 0x07, 0x13, 0xd0, 1 };
128         data_pkt[2] |= i;
129         *(ushort*)(data_pkt + 5) = dpi->x[i];
130         *(ushort*)(data_pkt + 7) = dpi->y[i];
131         // Save the RGB value for this setting too
132         data_pkt[9] = light->r[LED_MOUSE + N_MOUSE_ZONES + i];
133         data_pkt[10] = light->g[LED_MOUSE + N_MOUSE_ZONES + i];
134         data_pkt[11] = light->b[LED_MOUSE + N_MOUSE_ZONES + i];
135         if(!usbsend(kb, data_pkt, 1))
136             return -1;
137     }
138
139     // Send settings
140     uchar data_pkt[4][MSG_SIZE] = {
141         { 0x07, 0x13, 0x05, 1, dpi->enabled },
142         { 0x07, 0x13, 0x02, 1, dpi->current },
143         { 0x07, 0x13, 0x03, 1, dpi->lift },
144         { 0x07, 0x13, 0x04, 1, dpi->snap, 0x05 }
145     };
```

```
146     if(!usbsend(kb, data_pkt[0], 4))
147         return -2;
148     // Finished
149     return 0;
150 }
```

Here is the caller graph for this function:



8.15.1.8 int updatedpi (usbdevice * *kb*, int *force*)

Definition at line 89 of file dpi.c.

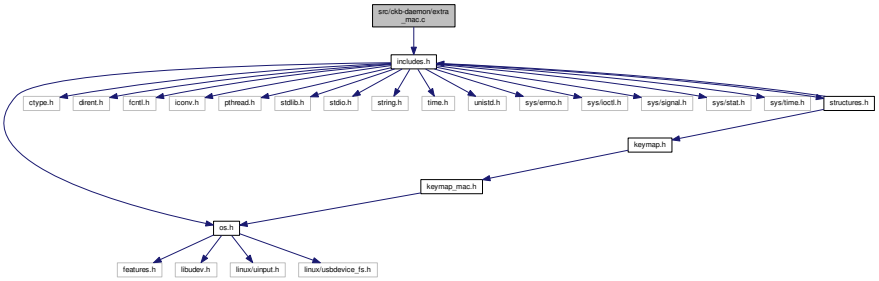
References usbdevice::active, dpiset::current, usbprofile::currentmode, usbmode::dpi, DPI_COUNT, dpiset::enabled, dpiset::forceupdate, usbprofile::lastdpi, dpiset::lift, MSG_SIZE, usbdevice::profile, dpiset::snap, usbsend, dpiset::x, and dpiset::y.

```
89                                      {
90     if(!kb->active)
91         return 0;
92     dpiset* lastdpi = &kb->profile->lastdpi;
93     dpiset* newdpi = &kb->profile->currentmode->dpi;
94     // Don't do anything if the settings haven't changed
95     if(!force && !lastdpi->forceupdate && !newdpi->forceupdate
96             && !memcmp(lastdpi, newdpi, sizeof(dpiset)))
97         return 0;
98     lastdpi->forceupdate = newdpi->forceupdate = 0;
99
100     // Send X/Y DPIs
101     for(int i = 0; i < DPI_COUNT; i++){
102         uchar data_pkt[MSG_SIZE] = { 0x07, 0x13, 0xd0, 0 };
103         data_pkt[2] |= i;
104         *(ushort*)(data_pkt + 5) = newdpi->x[i];
105         *(ushort*)(data_pkt + 7) = newdpi->y[i];
106         if(!usbsend(kb, data_pkt, 1))
107             return -1;
108     }
109
110     // Send settings
111     uchar data_pkt[4][MSG_SIZE] = {
112         { 0x07, 0x13, 0x05, 0, newdpi->enabled },
113         { 0x07, 0x13, 0x02, 0, newdpi->current },
114         { 0x07, 0x13, 0x03, 0, newdpi->lift },
115         { 0x07, 0x13, 0x04, 0, newdpi->snap, 0x05 }
116     };
117     if(!usbsend(kb, data_pkt[0], 4))
118         return -2;
119     // Finished
120     memcpy(lastdpi, newdpi, sizeof(dpiset));
121     return 0;
122 }
```

8.16 src/ckb-daemon/extra_mac.c File Reference

```
#include "includes.h"
```

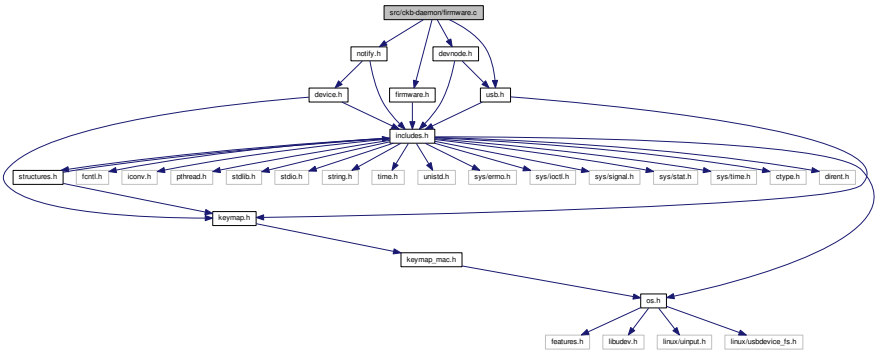
Include dependency graph for extra_mac.c:



8.17 src/ckb-daemon/firmware.c File Reference

```
#include "devnode.h"
#include "firmware.h"
#include "notify.h"
#include "usb.h"
```

Include dependency graph for firmware.c:



Macros

- #define FW_OK 0
- #define FW_NOFILE -1
- #define FW_WRONGDEV -2
- #define FW_USBFAIL -3
- #define FW_MAXSIZE (255 ∗ 256)

Functions

- int getfwversion (usbdevice ∗kb)
- int fwupdate (usbdevice ∗kb, const char ∗path, int nnumber)
- int cmd_fwupdate (usbdevice ∗kb, usbmode ∗dummy1, int nnumber, int dummy2, const char ∗path)

8.17.1 Macro Definition Documentation

8.17.1.1 #define FW_MAXSIZE (255 * 256)

Definition at line 51 of file firmware.c.

Referenced by fwupdate().

8.17.1.2 #define FW_NOFILE -1

Definition at line 7 of file firmware.c.

Referenced by cmd_fwupdate(), and fwupdate().

8.17.1.3 #define FW_OK 0

Definition at line 6 of file firmware.c.

Referenced by cmd_fwupdate(), and fwupdate().

8.17.1.4 #define FW_USBFAIL -3

Definition at line 9 of file firmware.c.

Referenced by cmd_fwupdate(), and fwupdate().

8.17.1.5 #define FW_WRONGDEV -2

Definition at line 8 of file firmware.c.

Referenced by cmd_fwupdate(), and fwupdate().

8.17.2 Function Documentation

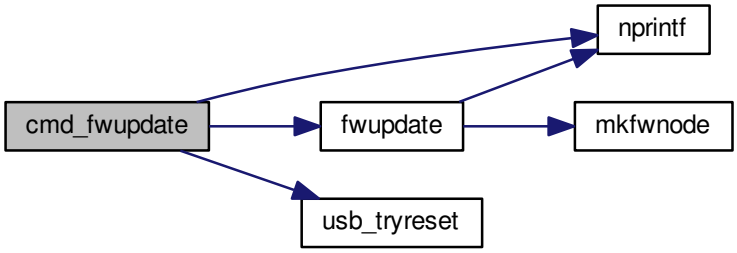
8.17.2.1 int cmd_fwupdate (usbdevice * *kb*, usbmode * *dummy1*, int *nnumber*, int *dummy2*, const char * *path*)

Definition at line 154 of file firmware.c.

References FEAT_FWUPDATE, FW_NOFILE, FW_OK, FW_USBFAIL, FW_WRONGDEV, fwupdate(), HAS_FEATURES, nprintf(), and usb_tryreset().

```
154                                                                                 {
155     if(!HAS_FEATURES(kb, FEAT_FWUPDATE))
156         return 0;
157     // Update the firmware
158     int ret = fwupdate(kb, path, nnumber);
159     while(ret == FW_USBFAIL){
160         // Try to reset the device if it fails
161         if(usb_tryreset(kb))
162             break;
163         ret = fwupdate(kb, path, nnumber);
164     }
165     switch(ret){
166     case FW_OK:
167         nprintf(kb, nnumber, 0, "fwupdate %s ok\n", path);
168         break;
169     case FW_NOFILE:
170     case FW_WRONGDEV:
171         nprintf(kb, nnumber, 0, "fwupdate %s invalid\n", path);
172         break;
173     case FW_USBFAIL:
174         nprintf(kb, nnumber, 0, "fwupdate %s fail\n", path);
175         return -1;
176     }
177     return 0;
178 }
```

Here is the call graph for this function:



8.17.2.2 int fwupdate (usbdevice * *kb*, const char * *path*, int *nnumber*)

Definition at line 55 of file firmware.c.

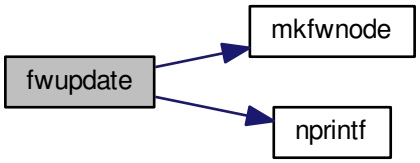
References ckb_err, ckb_info, FW_MAXSIZE, FW_NOFILE, FW_OK, FW_USBFAIL, FW_WRONGDEV, usbdevice::fwversion, mkfwnode(), MSG_SIZE, nprintf(), usbdevice::product, usbdevice::usbdelay, usbsend, and usbdevice::vendor.

Referenced by cmd_fwupdate().

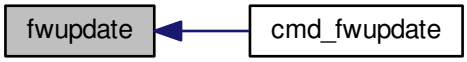
```
55                                                      {
56     // Read the firmware from the given path
57     char* fwdata = calloc(1, FW_MAXSIZE + 256);
58     int fd = open(path, O_RDONLY);
59     if(fd == -1){
60         ckb_err("Failed to open firmware file %s: %s\n", path, strerror(errno));
61         return FW_NOFILE;
62     }
63     ssize_t length = read(fd, fwdata, FW_MAXSIZE + 1);
64     if(length <= 0x108 || length > FW_MAXSIZE){
65         ckb_err("Failed to read firmware file %s: %s\n", path, length <= 0 ? strerror(errno) : "
     Wrong size");
66         close(fd);
67         return FW_NOFILE;
68     }
69     close(fd);
70 
71     short vendor, product, version;
72     // Copy the vendor ID, product ID, and version from the firmware file
73     memcpy(&vendor, fwdata + 0x102, 2);
74     memcpy(&product, fwdata + 0x104, 2);
75     memcpy(&version, fwdata + 0x106, 2);
76     // Check against the actual device
77     if(vendor != kb->vendor || product != kb->product){
78         ckb_err("Firmware file %s doesn't match device (V: %04x P: %04x)\n", path, vendor, product);
79         return FW_WRONGDEV;
80     }
81     ckb_info("Loading firmware version %04x from %s\n", version, path);
82     nprintf(kb, nnumber, 0, "fwupdate %s 0/%d\n", path, (int)length);
83     // Force the device to 10ms delay (we need to deliver packets very slowly to make sure it doesn't get
     overwhelmed)
84     kb->usbdelay = 10;
85     // Send the firmware messages (256 bytes at a time)
86     uchar data_pkt[7][MSG_SIZE] = {
87         { 0x07, 0x0c, 0xf0, 0x01, 0 },
88         { 0x07, 0x0d, 0xf0, 0 },
89         { 0x7f, 0x01, 0x3c, 0 },
90         { 0x7f, 0x02, 0x3c, 0 },
91         { 0x7f, 0x03, 0x3c, 0 },
92         { 0x7f, 0x04, 0x3c, 0 },
93         { 0x7f, 0x05, 0x10, 0 }
94     };
95     int output = 0, last = 0;
96     int index = 0;
```

```
97      while(output < length){
98          int npackets = 1;
99          // Packet 1: data position
100          data_pkt[1][6] = index++;
101          while(output < length){
102              npackets++;
103              if(npackets != 6){
104                  // Packets 2-5: 60 bytes of data
105                  memcpy(data_pkt[npackets] + 4, fwdata + output, 60);
106                  last = output;
107                  output += 60;
108              } else {
109                  // Packet 6: 16 bytes
110                  memcpy(data_pkt[npackets] + 4, fwdata + output, 16);
111                  last = output;
112                  output += 16;
113                  break;
114              }
115          }
116          if(index == 1){
117              if(!usbsend(kb, data_pkt[0], 1)){
118                  ckb_err("Firmware update failed\n");
119                  return FW_USBFAIL;
120              }
121              // The above packet can take a lot longer to process, so wait for a while
122              sleep(3);
123              if(!usbsend(kb, data_pkt[2], npackets - 1)){
124                  ckb_err("Firmware update failed\n");
125                  return FW_USBFAIL;
126              }
127          } else {
128              // If the output ends here, set the length byte appropriately
129              if(output >= length)
130                  data_pkt[npackets][2] = length - last;
131              if(!usbsend(kb, data_pkt[1], npackets)){
132                  ckb_err("Firmware update failed\n");
133                  return FW_USBFAIL;
134              }
135          }
136          nprintf(kb, nnumber, 0, "fwupdate %s %d/%d\n", path, output, (int)length);
137      }
138      // Send the final pair of messages
139      uchar data_pkt2[2][MSG_SIZE] = {
140          { 0x07, 0x0d, 0xf0, 0x00, 0x00, 0x00, index },
141          { 0x07, 0x02, 0xf0, 0 }
142      };
143      if(!usbsend(kb, data_pkt2[0], 2)){
144          ckb_err("Firmware update failed\n");
145          return FW_USBFAIL;
146      }
147      // Updated successfully
148      kb->fwversion = version;
149      mkfwnode(kb);
150      ckb_info("Firmware update complete\n");
151      return FW_OK;
152  }
```

Here is the call graph for this function:



Here is the caller graph for this function:



8.17.2.3 int getfwversion (usbdevice * *kb*)

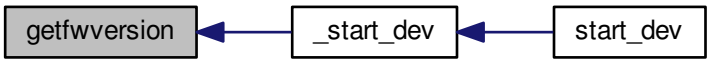
Definition at line 11 of file firmware.c.

References ckb_err, ckb_warn, FEAT_POLLRATE, usbdevice::features, usbdevice::fwversion, MSG_SIZE, usbdevice::pollrate, usbdevice::product, usbrecv, and usbdevice::vendor.

Referenced by _start_dev().

```
11                                {
12      // Ask board for firmware info
13      uchar data_pkt[MSG_SIZE] = { 0x0e, 0x01, 0 };
14      uchar in_pkt[MSG_SIZE];
15      if(!usbrecv(kb, data_pkt, in_pkt))
16          return -1;
17      if(in_pkt[0] != 0x0e || in_pkt[1] != 0x01){
18          ckb_err("Bad input header\n");
19          return -1;
20      }
21      short vendor, product, version, bootloader;
22      // Copy the vendor ID, product ID, version, and poll rate from the firmware data
23      memcpy(&version, in_pkt + 8, 2);
24      memcpy(&bootloader, in_pkt + 10, 2);
25      memcpy(&vendor, in_pkt + 12, 2);
26      memcpy(&product, in_pkt + 14, 2);
27      char poll = in_pkt[16];
28      if(poll <= 0){
29          poll = -1;
30          kb->features &= ~FEAT_POLLRATE;
31      }
32      // Print a warning if the message didn't match the expected data
33      if(vendor != kb->vendor)
34          ckb_warn("Got vendor ID %04x (expected %04x)\n", vendor, kb->
    vendor);
35      if(product != kb->product)
36          ckb_warn("Got product ID %04x (expected %04x)\n", product, kb->
    product);
37      // Set firmware version and poll rate
38      if(version == 0 || bootloader == 0){
39          // Needs firmware update
40          kb->fwversion = 0;
41          kb->pollrate = -1;
42      } else {
43          if(version != kb->fwversion && kb->fwversion != 0)
44              ckb_warn("Got firmware version %04x (expected %04x)\n", version, kb->
    fwversion);
45          kb->fwversion = version;
46          kb->pollrate = poll;
47      }
48      return 0;
49  }
```

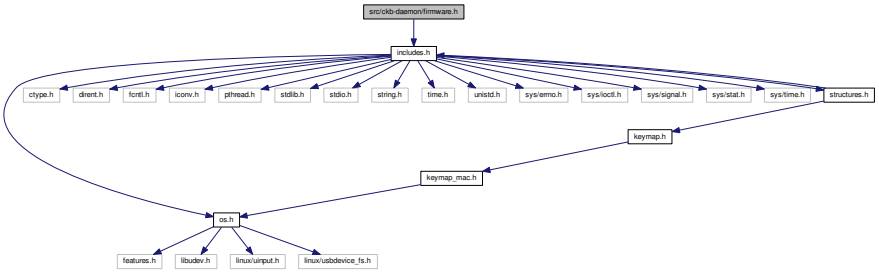
Here is the caller graph for this function:



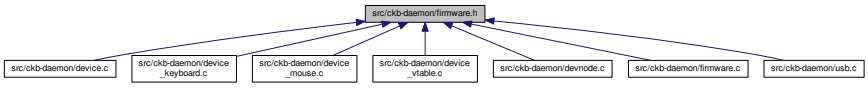
8.18 src/ckb-daemon/firmware.h File Reference

```
#include "includes.h"
```

Include dependency graph for firmware.h:



This graph shows which files directly or indirectly include this file:



Functions

- int getfwversion (usbdevice ∗kb)
- int cmd_fwupdate (usbdevice ∗kb, usbmode ∗dummy1, int nnumber, int dummy2, const char ∗path)

8.18.1 Function Documentation

8.18.1.1 int cmd_fwupdate (usbdevice ∗ *kb,* usbmode ∗ *dummy1,* int *nnumber,* int *dummy2,* const char ∗ *path*)

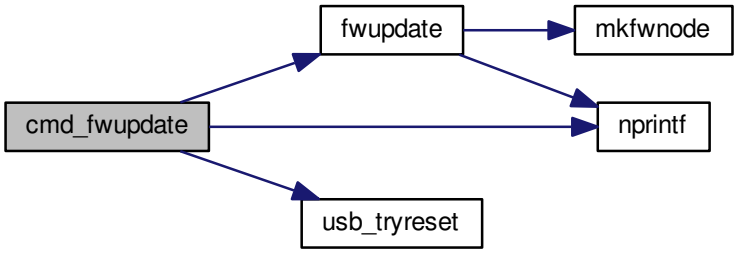
Definition at line 154 of file firmware.c.

References FEAT_FWUPDATE, FW_NOFILE, FW_OK, FW_USBFAIL, FW_WRONGDEV, fwupdate(), HAS_FEATURES, nprintf(), and usb_tryreset().

```
154                                                                                 {
155     if(!HAS_FEATURES(kb, FEAT_FWUPDATE))
156         return 0;
```

```
157     // Update the firmware
158     int ret = fwupdate(kb, path, nnumber);
159     while(ret == FW_USBFAIL){
160         // Try to reset the device if it fails
161         if(usb_tryreset(kb))
162             break;
163         ret = fwupdate(kb, path, nnumber);
164     }
165     switch(ret){
166     case FW_OK:
167         nprintf(kb, nnumber, 0, "fwupdate %s ok\n", path);
168         break;
169     case FW_NOFILE:
170     case FW_WRONGDEV:
171         nprintf(kb, nnumber, 0, "fwupdate %s invalid\n", path);
172         break;
173     case FW_USBFAIL:
174         nprintf(kb, nnumber, 0, "fwupdate %s fail\n", path);
175         return -1;
176     }
177     return 0;
178 }
```

Here is the call graph for this function:



8.18.1.2 int getfwversion (usbdevice * *kb*)

Definition at line 11 of file firmware.c.

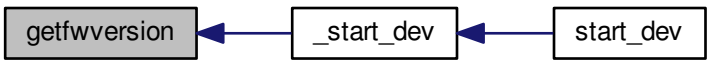
References ckb_err, ckb_warn, FEAT_POLLRATE, usbdevice::features, usbdevice::fwversion, MSG_SIZE, usbdevice::pollrate, usbdevice::product, usbrecv, and usbdevice::vendor.

Referenced by _start_dev().

```
11                                  {
12     // Ask board for firmware info
13     uchar data_pkt[MSG_SIZE] = { 0x0e, 0x01, 0 };
14     uchar in_pkt[MSG_SIZE];
15     if(!usbrecv(kb, data_pkt, in_pkt))
16         return -1;
17     if(in_pkt[0] != 0x0e || in_pkt[1] != 0x01){
18         ckb_err("Bad input header\n");
19         return -1;
20     }
21     short vendor, product, version, bootloader;
22     // Copy the vendor ID, product ID, version, and poll rate from the firmware data
23     memcpy(&version, in_pkt + 8, 2);
24     memcpy(&bootloader, in_pkt + 10, 2);
25     memcpy(&vendor, in_pkt + 12, 2);
26     memcpy(&product, in_pkt + 14, 2);
27     char poll = in_pkt[16];
28     if(poll <= 0){
29         poll = -1;
30         kb->features &= ~FEAT_POLLRATE;
31     }
```

```
32     // Print a warning if the message didn't match the expected data
33     if(vendor != kb->vendor)
34         ckb_warn("Got vendor ID %04x (expected %04x)\n", vendor, kb->
       vendor);
35     if(product != kb->product)
36         ckb_warn("Got product ID %04x (expected %04x)\n", product, kb->
       product);
37     // Set firmware version and poll rate
38     if(version == 0 || bootloader == 0){
39         // Needs firmware update
40         kb->fwversion = 0;
41         kb->pollrate = -1;
42     } else {
43         if(version != kb->fwversion && kb->fwversion != 0)
44             ckb_warn("Got firmware version %04x (expected %04x)\n", version, kb->
       fwversion);
45         kb->fwversion = version;
46         kb->pollrate = poll;
47     }
48     return 0;
49 }
```

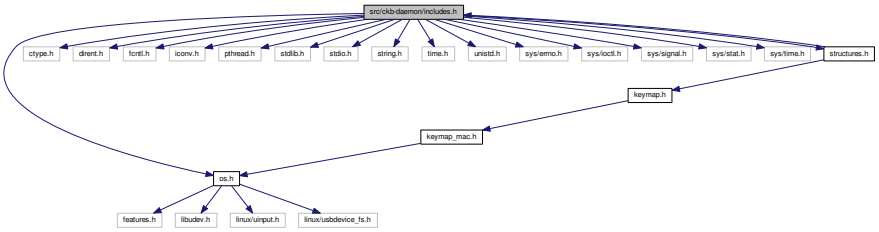
Here is the caller graph for this function:



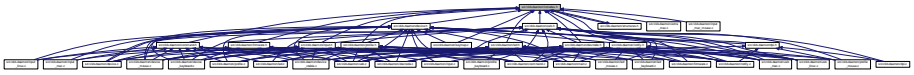
8.19 src/ckb-daemon/includes.h File Reference

```
#include "os.h"
#include <ctype.h>
#include <dirent.h>
#include <fcntl.h>
#include <iconv.h>
#include <pthread.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <time.h>
#include <unistd.h>
#include <sys/errno.h>
#include <sys/ioctl.h>
#include <sys/signal.h>
#include <sys/stat.h>
#include <sys/time.h>
#include "structures.h"
```

Include dependency graph for includes.h:



This graph shows which files directly or indirectly include this file:



Macros

- #define INDEX_OF(entry, array) (int)(entry - array)
- #define ckb_s_out stdout
- #define ckb_s_err stdout
- #define __FILE_NOPATH__ (strrchr(__FILE__, '/') ? strrchr(__FILE__, '/') + 1 : __FILE__)
- #define ckb_fatal_nofile(fmt, args...) fprintf(ckb_s_err, "[F] " fmt, ## args)
- #define ckb_fatal_fn(fmt, file, line, args...) fprintf(ckb_s_err, "[F] %s (via %s:%d): " fmt, __func__, file, line, ## args)
- #define ckb_fatal(fmt, args...) fprintf(ckb_s_err, "[F] %s (%s:%d): " fmt, __func__, __FILE_NOPATH__, __LINE__, ## args)
- #define ckb_err_nofile(fmt, args...) fprintf(ckb_s_err, "[E] " fmt, ## args)
- #define ckb_err_fn(fmt, file, line, args...) fprintf(ckb_s_err, "[E] %s (via %s:%d): " fmt, __func__, file, line, ## args)
- #define ckb_err(fmt, args...) fprintf(ckb_s_err, "[E] %s (%s:%d): " fmt, __func__, __FILE_NOPATH__, __LINE__, ## args)
- #define ckb_warn_nofile(fmt, args...) fprintf(ckb_s_out, "[W] " fmt, ## args)
- #define ckb_warn_fn(fmt, file, line, args...) fprintf(ckb_s_out, "[W] %s (via %s:%d): " fmt, __func__, file, line, ## args)
- #define ckb_warn(fmt, args...) fprintf(ckb_s_out, "[W] %s (%s:%d): " fmt, __func__, __FILE_NOPATH__, __LINE__, ## args)
- #define ckb_info_nofile(fmt, args...) fprintf(ckb_s_out, "[I] " fmt, ## args)
- #define ckb_info_fn(fmt, file, line, args...) fprintf(ckb_s_out, "[I] " fmt, ## args)
- #define ckb_info(fmt, args...) fprintf(ckb_s_out, "[I] " fmt, ## args)
- #define timespec_gt(left, right) ((left).tv_sec > (right).tv_sec || ((left).tv_sec == (right).tv_sec && (left).tv_nsec > (right).tv_nsec))
- #define timespec_eq(left, right) ((left).tv_sec == (right).tv_sec && (left).tv_nsec == (right).tv_nsec)
- #define timespec_ge(left, right) ((left).tv_sec > (right).tv_sec || ((left).tv_sec == (right).tv_sec && (left).tv_nsec >= (right).tv_nsec))
- #define timespec_lt(left, right) (!timespec_ge(left, right))
- #define timespec_le(left, right) (!timespec_gt(left, right))

Typedefs

- typedef unsigned char uchar
- typedef unsigned short ushort

Functions

- void timespec_add (struct timespec *timespec, long nanoseconds)

8.19.1 Macro Definition Documentation

8.19.1.1 #define __FILE_NOPATH__ (strrchr(__FILE__, '/') ? strrchr(__FILE__, '/') + 1 : __FILE__)

Definition at line 40 of file includes.h.

8.19.1.2 #define ckb_err(*fmt, args...*) fprintf(ckb_s_err, "[E] %s (%s:%d): " fmt, __func__, __FILE_NOPATH__, __LINE__, ## args)

Definition at line 49 of file includes.h.

Referenced by _mkdevpath(), fwupdate(), getfwversion(), loaddpi(), loadrgb_kb(), loadrgb_mouse(), os_sendindicators(), os_setupusb(), restart(), setupusb(), uinputopen(), usb_tryreset(), and usbadd().

8.19.1.3 #define ckb_err_fn(*fmt, file, line, args...*) fprintf(ckb_s_err, "[E] %s (via %s:%d): " fmt, __func__, file, line, ## args)

Definition at line 48 of file includes.h.

Referenced by _nk95cmd(), _usbrecv(), os_usbrecv(), and os_usbsend().

8.19.1.4 #define ckb_err_nofile(*fmt, args...*) fprintf(ckb_s_err, "[E] " fmt, ## args)

Definition at line 47 of file includes.h.

8.19.1.5 #define ckb_fatal(*fmt, args...*) fprintf(ckb_s_err, "[F] %s (%s:%d): " fmt, __func__, __FILE_NOPATH__, __LINE__, ## args)

Definition at line 46 of file includes.h.

Referenced by usbmain().

8.19.1.6 #define ckb_fatal_fn(*fmt, file, line, args...*) fprintf(ckb_s_err, "[F] %s (via %s:%d): " fmt, __func__, file, line, ## args)

Definition at line 45 of file includes.h.

8.19.1.7 #define ckb_fatal_nofile(*fmt, args...*) fprintf(ckb_s_err, "[F] " fmt, ## args)

Definition at line 44 of file includes.h.

Referenced by main().

8.19.1.8 #define ckb_info(*fmt, args...*) fprintf(ckb_s_out, "[I] " fmt, ## args)

Definition at line 55 of file includes.h.

Referenced by _setupusb(), _start_dev(), closeusb(), cmd_restart(), fwupdate(), main(), os_inputmain(), os_setupusb(), quitWithLock(), rmdevpath(), and usb_tryreset().

8.19.1.9 #define ckb_info_fn(*fmt, file, line, args...*) fprintf(ckb_s_out, "[I] " fmt, ## args)

Definition at line 54 of file includes.h.

8.19.1.10 #define ckb_info_nofile(*fmt, args...*) fprintf(ckb_s_out, "[I] " fmt, ## args)

Definition at line 53 of file includes.h.

Referenced by main().

8.19.1.11 #define ckb_s_err stdout

Definition at line 36 of file includes.h.

8.19.1.12 #define ckb_s_out stdout

Definition at line 35 of file includes.h.

8.19.1.13 #define ckb_warn(*fmt, args...*) fprintf(ckb_s_out, "[W] %s (%s:%d): " fmt, __func__, __FILE_NOPATH__, __LINE__, ## args)

Definition at line 52 of file includes.h.

Referenced by _mkdevpath(), _mknotifynode(), _start_dev(), _updateconnected(), getfwversion(), hid_kb_translate(), isync(), mkfwnode(), os_inputclose(), os_keypress(), os_mousemove(), os_setupusb(), readlines(), rmdevpath(), uinputopen(), and usbmain().

8.19.1.14 #define ckb_warn_fn(*fmt, file, line, args...*) fprintf(ckb_s_out, "[W] %s (via %s:%d): " fmt, __func__, file, line, ## args)

Definition at line 51 of file includes.h.

Referenced by os_usbrecv(), and os_usbsend().

8.19.1.15 #define ckb_warn_nofile(*fmt, args...*) fprintf(ckb_s_out, "[W] " fmt, ## args)

Definition at line 50 of file includes.h.

Referenced by main().

8.19.1.16 #define INDEX_OF(*entry, array*) (int)(entry - array)

Definition at line 27 of file includes.h.

Referenced by _mkdevpath(), _mknotifynode(), _rmnotifynode(), _setupusb(), closeusb(), mkfwnode(), nprintf(), os_closeusb(), os_inputmain(), os_inputopen(), os_setupusb(), readcmd(), and rmdevpath().

8.19.1.17 #define timespec_eq(*left, right*) ((left).tv_sec == (right).tv_sec && (left).tv_nsec == (right).tv_nsec)

Definition at line 60 of file includes.h.

8.19.1.18 #define timespec_ge(*left, right*) ((left).tv_sec > (right).tv_sec || ((left).tv_sec == (right).tv_sec && (left).tv_nsec >= (right).tv_nsec))

Definition at line 61 of file includes.h.

8.19.1.19 #define timespec_gt(*left, right*) ((left).tv_sec > (right).tv_sec || ((left).tv_sec == (right).tv_sec && (left).tv_nsec > (right).tv_nsec))

Definition at line 59 of file includes.h.

8.19.1.20 #define timespec_le(*left, right*) (!timespec_gt(left, right))

Definition at line 63 of file includes.h.

8.19.1.21 #define timespec_lt(*left, right*) (!timespec_ge(left, right))

Definition at line 62 of file includes.h.

8.19.2 Typedef Documentation

8.19.2.1 typedef unsigned char uchar

Definition at line 24 of file includes.h.

8.19.2.2 typedef unsigned short ushort

Definition at line 25 of file includes.h.

8.19.3 Function Documentation

8.19.3.1 void timespec_add (struct timespec * *timespec,* long *nanoseconds*)

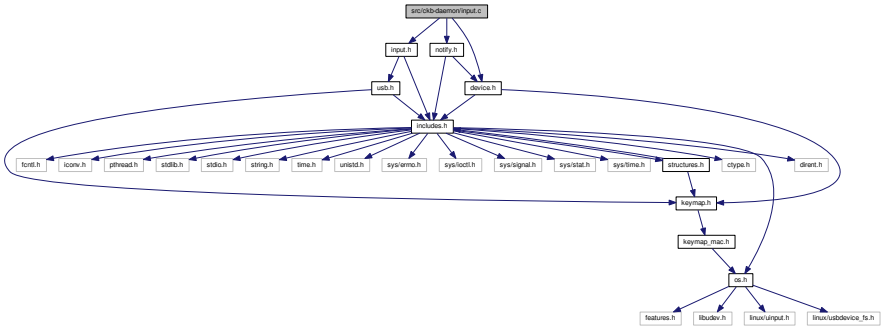
Definition at line 19 of file main.c.

```
19                                                          {
20     nanoseconds += timespec->tv_nsec;
21     timespec->tv_sec += nanoseconds / 1000000000;
22     timespec->tv_nsec = nanoseconds % 1000000000;
23 }
```

8.20 src/ckb-daemon/input.c File Reference

```
#include "device.h"
#include "input.h"
#include "notify.h"
```

Include dependency graph for input.c:



Macros

- #define IS_WHEEL(scan, kb) (((scan) == KEY_VOLUMEUP || (scan) == KEY_VOLUMEDOWN || (scan) == BTN_WHEELUP || (scan) == BTN_WHEELDOWN) && !IS_K65(kb))

Functions

- int macromask (const uchar *key1, const uchar *key2)
- static void inputupdate_keys (usbdevice *kb)
- void inputupdate (usbdevice *kb)
- void updateindicators_kb (usbdevice *kb, int force)
- void initbind (binding *bind)
- void freebind (binding *bind)
- void cmd_bind (usbdevice *kb, usbmode *mode, int dummy, int keyindex, const char *to)
- void cmd_unbind (usbdevice *kb, usbmode *mode, int dummy, int keyindex, const char *to)
- void cmd_rebind (usbdevice *kb, usbmode *mode, int dummy, int keyindex, const char *to)
- static void _cmd_macro (usbmode *mode, const char *keys, const char *assignment)
- void cmd_macro (usbdevice *kb, usbmode *mode, const int notifynumber, const char *keys, const char *assignment)

8.20.1 Macro Definition Documentation

8.20.1.1 #define IS_WHEEL(*scan, kb*) (((scan) == KEY_VOLUMEUP || (scan) == KEY_VOLUMEDOWN || (scan) == BTN_WHEELUP || (scan) == BTN_WHEELDOWN) && !IS_K65(kb))

Referenced by inputupdate_keys().

8.20.2 Function Documentation

8.20.2.1 static void _cmd_macro (usbmode * *mode,* const char * *keys,* const char * *assignment*) [static]

Definition at line 226 of file input.c.

References keymacro::actioncount, keymacro::actions, usbmode::bind, keymacro::combo, macroaction::down, keymap, MACRO_MAX, binding::macrocap, binding::macrocount, binding::macros, N_KEYBYTES_INPUT, N_KEYS_INPUT, macroaction::scan, key::scan, and SET_KEYBIT.

Referenced by cmd_macro().

```
226                                                                                 {
227     binding* bind = &mode->bind;
228     if(!keys && !assignment){
229         // Null strings = "macro clear" -> erase the whole thing
230         for(int i = 0; i < bind->macrocount; i++)
231             free(bind->macros[i].actions);
232         bind->macrocount = 0;
233         return;
234     }
235     if(bind->macrocount >= MACRO_MAX)
236         return;
237     // Create a key macro
238     keymacro macro;
239     memset(&macro, 0, sizeof(macro));
240     // Scan the left side for key names, separated by +
241     int empty = 1;
242     int left = strlen(keys), right = strlen(assignment);
243     int position = 0, field = 0;
244     char keyname[12];
245     while(position < left && sscanf(keys + position, "%10[^+]%n", keyname, &field) == 1){
246         int keycode;
247         if((sscanf(keyname, "#%d", &keycode) && keycode >= 0 && keycode <
    N_KEYS_INPUT)
248                 || (sscanf(keyname, "#x%x", &keycode) && keycode >= 0 && keycode <
    N_KEYS_INPUT)){
249             // Set a key numerically
250             SET_KEYBIT(macro.combo, keycode);
251             empty = 0;
252         } else {
253             // Find this key in the keymap
254             for(unsigned i = 0; i < N_KEYS_INPUT; i++){
255                 if(keymap[i].name && !strcmp(keyname, keymap[i].name)){
256                     macro.combo[i / 8] |= 1 << (i % 8);
257                     empty = 0;
258                     break;
259                 }
260             }
261         }
262         if(keys[position += field] == '+')
263             position++;
264     }
265     if(empty)
266         return;
267     // Count the number of actions (comma separated)
268     int count = 1;
269     for(const char* c = assignment; *c != 0; c++){
270         if(*c == ',')
271             count++;
272     }
273     // Allocate a buffer for them
274     macro.actions = calloc(count, sizeof(macroaction));
275     macro.actioncount = 0;
276     // Scan the actions
277     position = 0;
278     field = 0;
279     while(position < right && sscanf(assignment + position, "%11[^,]%n", keyname, &field) == 1){
280         if(!strcmp(keyname, "clear"))
281             break;
282         int down = (keyname[0] == '+');
283         if(down || keyname[0] == '-'){
284             int keycode;
285             if((sscanf(keyname + 1, "#%d", &keycode) && keycode >= 0 && keycode < N_KEYS_INPUT)
286                     || (sscanf(keyname + 1, "#x%x", &keycode) && keycode >= 0 && keycode <
    N_KEYS_INPUT)){
287                 // Set a key numerically
288                 macro.actions[macro.actioncount].scan =
    keymap[keycode].scan;
289                 macro.actions[macro.actioncount].down = down;
290                 macro.actioncount++;
291             } else {
292                 // Find this key in the keymap
293                 for(unsigned i = 0; i < N_KEYS_INPUT; i++){
294                     if(keymap[i].name && !strcmp(keyname + 1, keymap[i].name)){
295                         macro.actions[macro.actioncount].scan =
    keymap[i].scan;
296                         macro.actions[macro.actioncount].down = down;
297                         macro.actioncount++;
298                         break;
299                     }
300                 }
301             }
302         }
303         if(assignment[position += field] == ',')
304             position++;
305     }
306
307     // See if there's already a macro with this trigger
```

```
308     keymacro* macros = bind->macros;
309     for(int i = 0; i < bind->macrocount; i++){
310         if(!memcmp(macros[i].combo, macro.combo, N_KEYBYTES_INPUT)){
311             free(macros[i].actions);
312             // If the new macro has no actions, erase the existing one
313             if(!macro.actioncount){
314                 for(int j = i + 1; j < bind->macrocount; j++)
315                     memcpy(macros + j - 1, macros + j, sizeof(keymacro));
316                 bind->macrocount--;
317             } else
318                 // If there are actions, replace the existing with the new
319                 memcpy(macros + i, &macro, sizeof(keymacro));
320             return;
321         }
322     }
323
324     // Add the macro to the device settings if not empty
325     if(macro.actioncount < 1)
326         return;
327     memcpy(bind->macros + (bind->macrocount++), &macro, sizeof(
    keymacro));
328     if(bind->macrocount >= bind->macrocap)
329         bind->macros = realloc(bind->macros, (bind->macrocap += 16) * sizeof(
    keymacro));
330 }
```

Here is the caller graph for this function:

8.20.2.2 void cmd_bind (usbdevice * *kb*, usbmode * *mode*, int *dummy*, int *keyindex*, const char * *to*)

Definition at line 188 of file input.c.

References binding::base, usbmode::bind, imutex, keymap, N_KEYS_INPUT, and key::scan.

```
188                                                                                 {
189     if(keyindex >= N_KEYS_INPUT)
190         return;
191     // Find the key to bind to
192     int tocode = 0;
193     if(sscanf(to, "#x%ux", &tocode) != 1 && sscanf(to, "#%u", &tocode) == 1 && tocode <
    N_KEYS_INPUT){
194         pthread_mutex_lock(imutex(kb));
195         mode->bind.base[keyindex] = tocode;
196         pthread_mutex_unlock(imutex(kb));
197         return;
198     }
199     // If not numeric, look it up
200     for(int i = 0; i < N_KEYS_INPUT; i++){
201         if(keymap[i].name && !strcmp(to, keymap[i].name)){
202             pthread_mutex_lock(imutex(kb));
203             mode->bind.base[keyindex] = keymap[i].scan;
204             pthread_mutex_unlock(imutex(kb));
205             return;
206         }
207     }
208 }
```

8.20.2.3 void cmd_macro (usbdevice * *kb*, usbmode * *mode*, const int *notifynumber*, const char * *keys*, const char * *assignment*)

Definition at line 332 of file input.c.

References _cmd_macro(), and imutex.

```
332
        {
333     pthread_mutex_lock(imutex(kb));
334     _cmd_macro(mode, keys, assignment);
335     pthread_mutex_unlock(imutex(kb));
336 }
```

Here is the call graph for this function:

8.20.2.4 void cmd_rebind (usbdevice * *kb*, usbmode * *mode*, int *dummy*, int *keyindex*, const char * *to*)

Definition at line 218 of file input.c.

References binding::base, usbmode::bind, imutex, keymap, N_KEYS_INPUT, and key::scan.

```
218                                                                                 {
219     if(keyindex >= N_KEYS_INPUT)
220         return;
221     pthread_mutex_lock(imutex(kb));
222     mode->bind.base[keyindex] = keymap[keyindex].scan;
223     pthread_mutex_unlock(imutex(kb));
224 }
```

8.20.2.5 void cmd_unbind (usbdevice * *kb*, usbmode * *mode*, int *dummy*, int *keyindex*, const char * *to*)

Definition at line 210 of file input.c.

References binding::base, usbmode::bind, imutex, KEY_UNBOUND, and N_KEYS_INPUT.

```
210                                                                                 {
211     if(keyindex >= N_KEYS_INPUT)
212         return;
213     pthread_mutex_lock(imutex(kb));
214     mode->bind.base[keyindex] = KEY_UNBOUND;
215     pthread_mutex_unlock(imutex(kb));
216 }
```

8.20.2.6 void freebind (binding * *bind*)

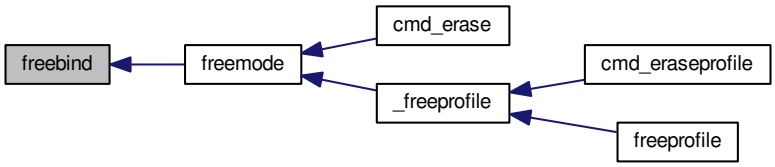
Definition at line 181 of file input.c.

References keymacro::actions, binding::macrocount, and binding::macros.

Referenced by freemode().

```
181                             {
182     for(int i = 0; i < bind->macrocount; i++)
183         free(bind->macros[i].actions);
184     free(bind->macros);
185     memset(bind, 0, sizeof(*bind));
186 }
```

Here is the caller graph for this function:



8.20.2.7 void initbind (binding * *bind*)

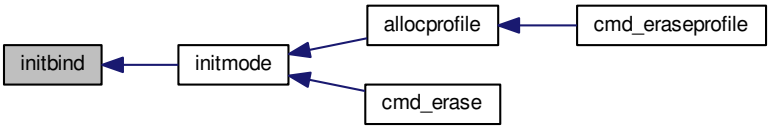
Definition at line 173 of file input.c.

References binding::base, keymap, binding::macrocap, binding::macrocount, binding::macros, N_KEYS_INPUT, and key::scan.

Referenced by initmode().

```
173                             {
174     for(int i = 0; i < N_KEYS_INPUT; i++)
175         bind->base[i] = keymap[i].scan;
176     bind->macros = calloc(32, sizeof(keymacro));
177     bind->macrocap = 32;
178     bind->macrocount = 0;
179 }
```

Here is the caller graph for this function:



8.20.2.8 void inputupdate (usbdevice * *kb*)

Definition at line 122 of file input.c.

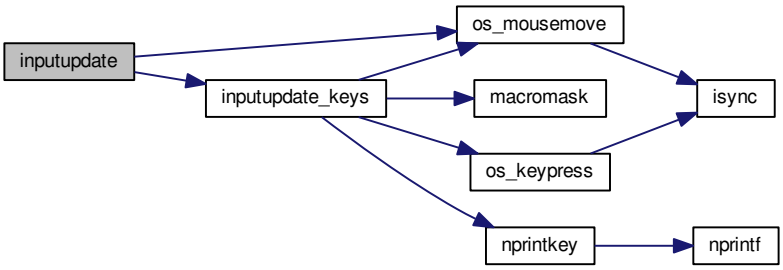
References usbdevice::input, inputupdate_keys(), os_mousemove(), usbdevice::profile, usbinput::rel_x, usbinput::rel_y, usbdevice::uinput_kb, and usbdevice::uinput_mouse.

Referenced by os_inputmain(), setactive_kb(), and setactive_mouse().

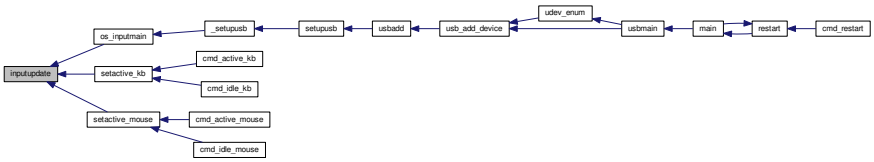
```
122                                 {
123 #ifdef OS_LINUX
124     if((!kb->uinput_kb || !kb->uinput_mouse)
125 #else
126     if(!kb->event
127 #endif
128             || !kb->profile)
129         return;
```

```
130     // Process key/button input
131     inputupdate_keys(kb);
132     // Process mouse movement
133     usbinput* input = &kb->input;
134     if(input->rel_x != 0 || input->rel_y != 0){
135         os_mousemove(kb, input->rel_x, input->rel_y);
136         input->rel_x = input->rel_y = 0;
137     }
138     // Finish up
139     memcpy(input->prevkeys, input->keys, N_KEYBYTES_INPUT);
140 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



8.20.2.9 static void inputupdate_keys (usbdevice * *kb*) [static]

Definition at line 15 of file input.c.

References keymacro::actioncount, keymacro::actions, usbdevice::active, binding::base, usbmode::bind, keymacro::combo, usbprofile::currentmode, usbdevice::delay, macroaction::down, usbdevice::input, IS_MOD, IS_WHEEL, keymap, usbinput::keys, binding::macrocount, macromask(), binding::macros, N_KEYBYTES_INPUT, N_KEYS_INPUT, usbmode::notify, nprintkey(), os_keypress(), os_mousemove(), OUTFIFO_MAX, usbinput::prevkeys, usbdevice::profile, macroaction::rel_x, macroaction::rel_y, macroaction::scan, key::scan, SCAN_SILENT, and keymacro::triggered.

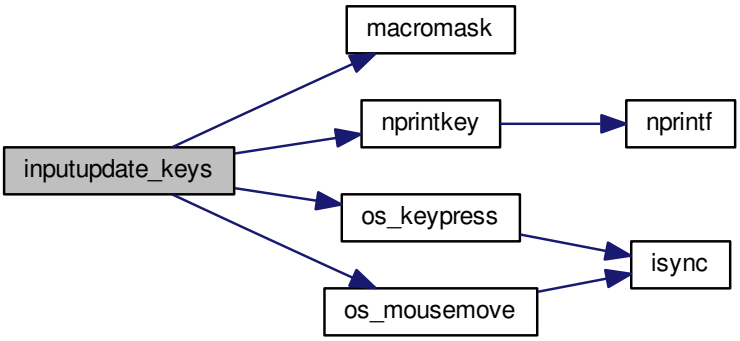
Referenced by inputupdate().

```
15                                          {
16     usbmode* mode = kb->profile->currentmode;
17     binding* bind = &mode->bind;
18     usbinput* input = &kb->input;
19     // Don't do anything if the state hasn't changed
20     if(!memcmp(input->prevkeys, input->keys, N_KEYBYTES_INPUT))
21         return;
22     // Look for macros matching the current state
23     int macrotrigger = 0;
24     if(kb->active){
25         for(int i = 0; i < bind->macrocount; i++){
```

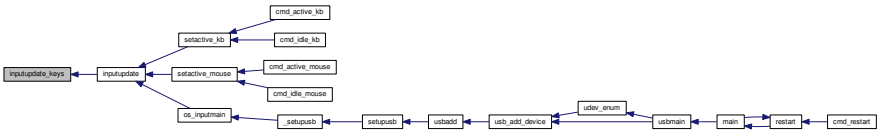
```
26              keymacro* macro = &bind->macros[i];
27              if(macromask(input->keys, macro->combo)){
28                  if(!macro->triggered){
29                      macrotrigger = 1;
30                      macro->triggered = 1;
31                      // Send events for each keypress in the macro
32                      for(int a = 0; a < macro->actioncount; a++){
33                          macroaction* action = macro->actions + a;
34                          if(action->rel_x != 0 || action->rel_y != 0)
35                              os_mousemove(kb, action->rel_x, action->
    rel_y);
36                          else {
37                              os_keypress(kb, action->scan, action->
    down);
38                              if (kb->delay) {
39                                  if (a > 200) usleep (100);
40                                  else if (a > 20) usleep(30);
41                              }
42                          }
43                      }
44                  }
45              } else {
46                  macro->triggered = 0;
47              }
48          }
49      }
50      // Make a list of keycodes to send. Rearrange them so that modifier keydowns always come first
51      // and modifier keyups always come last. This ensures that shortcut keys will register properly
52      // even if both keydown events happen at once.
53      // N_KEYS + 4 is used because the volume wheel generates keydowns and keyups at the same time
54      // (it's currently impossible to press all four at once, but safety first)
55      int events[N_KEYS_INPUT + 4];
56      int modcount = 0, keycount = 0, rmodcount = 0;
57      for(int byte = 0; byte < N_KEYBYTES_INPUT; byte++){
58          char oldb = input->prevkeys[byte], newb = input->keys[byte];
59          if(oldb == newb)
60              continue;
61          for(int bit = 0; bit < 8; bit++){
62              int keyindex = byte * 8 + bit;
63              if(keyindex >= N_KEYS_INPUT)
64                  break;
65              const key* map = keymap + keyindex;
66              int scancode = (kb->active) ? bind->base[keyindex] : map->
    scan;
67              char mask = 1 << bit;
68              char old = oldb & mask, new = newb & mask;
69              // If the key state changed, send it to the input device
70              if(old != new){
71                  // Don't echo a key press if a macro was triggered or if there's no scancode associated
72                  if(!macrotrigger && !(scancode & SCAN_SILENT)){
73                      if(IS_MOD(scancode)){
74                          if(new){
75                              // Modifier down: Add to the end of modifier keys
76                              for(int i = keycount + rmodcount; i > 0; i--)
77                                  events[modcount + i] = events[modcount + i - 1];
78                              // Add 1 to the scancode because A is zero on OSX
79                              // Positive code = keydown, negative code = keyup
80                              events[modcount++] = scancode + 1;
81                          } else {
82                              // Modifier up: Add to the end of everything
83                              events[modcount + keycount + rmodcount++] = -(scancode + 1);
84                          }
85                      } else {
86                          // Regular keypress: add to the end of regular keys
87                          for(int i = rmodcount; i > 0; i--)
88                              events[modcount + keycount + i] = events[modcount + keycount + i - 1];
89                          events[modcount + keycount++] = new ? (scancode + 1) : -(scancode + 1);
90                          // The volume wheel and the mouse wheel don't generate keyups, so create them
    automatically
91 #define IS_WHEEL(scan, kb)  (((scan) == KEY_VOLUMEUP || (scan) == KEY_VOLUMEDOWN || (scan) == BTN_WHEELUP
    || (scan) == BTN_WHEELDOWN) && !IS_K65(kb))
92                          if(new && IS_WHEEL(map->scan, kb)){
93                              for(int i = rmodcount; i > 0; i--)
94                                  events[modcount + keycount + i] = events[modcount + keycount + i - 1];
95                              events[modcount + keycount++] = -(scancode + 1);
96                              input->keys[byte] &= ~mask;
97                          }
98                      }
99                  }
100                 // Print notifications if desired
101                 if(kb->active){
102                     for(int notify = 0; notify < OUTFIFO_MAX; notify++){
103                         if(mode->notify[notify][byte] & mask){
104                             nprintkey(kb, notify, keyindex, new);
105                             // Wheels doesn't generate keyups
106                             if(new && IS_WHEEL(map->scan, kb))
107                                 nprintkey(kb, notify, keyindex, 0);
```

```
108                         }
109                     }
110                 }
111             }
112         }
113     }
114     // Process all queued keypresses
115     int totalkeys = modcount + keycount + rmodcount;
116     for(int i = 0; i < totalkeys; i++){
117         int scancode = events[i];
118         os_keypress(kb, (scancode < 0 ? -scancode : scancode) - 1, scancode > 0);
119     }
120 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



8.20.2.10 int macromask (const uchar * *key1*, const uchar * *key2*)

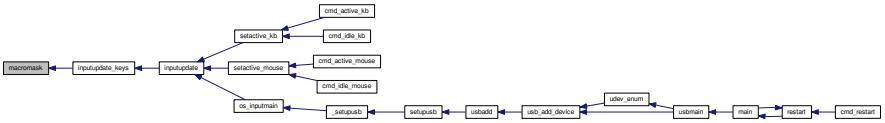
Definition at line 5 of file input.c.

References N_KEYBYTES_INPUT.

Referenced by inputupdate_keys().

```
5                                                   {
6     // Scan a macro against key input. Return 0 if any of them don't match
7     for(int i = 0; i < N_KEYBYTES_INPUT; i++){
8         // if((key1[i] & key2[i]) != key2[i])
9         if(key1[i] != key2[i])  // Changed to detect G-keys + modifiers
10             return 0;
11     }
12     return 1;
13 }
```

Here is the caller graph for this function:



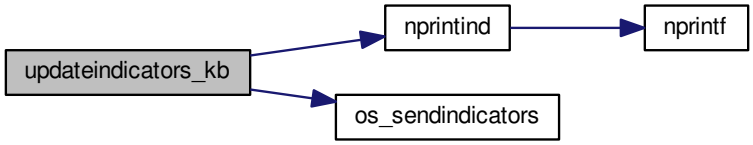
8.20.2.11 void updateindicators_kb (usbdevice * *kb*, int *force*)

Definition at line 142 of file input.c.

References usbdevice::active, usbprofile::currentmode, DELAY_SHORT, usbdevice::hw_ileds, usbdevice::hw_ileds_old, I_CAPS, I_NUM, I_SCROLL, usbdevice::ileds, usbmode::inotify, usbmode::ioff, usbmode::ion, nprintind(), os_sendindicators(), OUTFIFO_MAX, and usbdevice::profile.

```
142                                                 {
143     // Read current hardware indicator state (set externally)
144     uchar old = kb->ileds, hw_old = kb->hw_ileds_old;
145     uchar new = kb->hw_ileds, hw_new = new;
146     // Update them if needed
147     if(kb->active){
148         usbmode* mode = kb->profile->currentmode;
149         new = (new & ~mode->ioff) | mode->ion;
150     }
151     kb->ileds = new;
152     kb->hw_ileds_old = hw_new;
153     if(old != new || force){
154         DELAY_SHORT(kb);
155         os_sendindicators(kb);
156     }
157     // Print notifications if desired
158     if(!kb->active)
159         return;
160     usbmode* mode = kb->profile->currentmode;
161     uchar indicators[] = { I_NUM, I_CAPS, I_SCROLL };
162     for(unsigned i = 0; i < sizeof(indicators) / sizeof(uchar); i++){
163         uchar mask = indicators[i];
164         if((hw_old & mask) == (hw_new & mask))
165             continue;
166         for(int notify = 0; notify < OUTFIFO_MAX; notify++){
167             if(mode->inotify[notify] & mask)
168                 nprintind(kb, notify, mask, hw_new & mask);
169         }
170     }
171 }
```

Here is the call graph for this function:

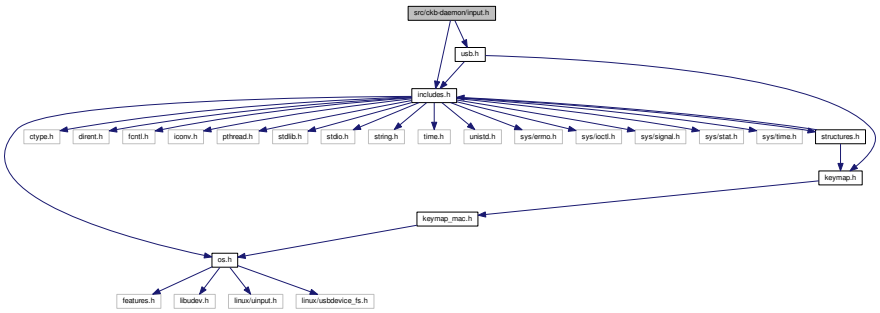


8.21 src/ckb-daemon/input.h File Reference

```
#include "includes.h"
```

```
#include "usb.h"
```

Include dependency graph for input.h:



This graph shows which files directly or indirectly include this file:

Macros

- #define IS_MOD(s) ((s) == KEY_CAPSLOCK || (s) == KEY_NUMLOCK || (s) == KEY_SCROLLLOCK || (s) == KEY_LEFTSHIFT || (s) == KEY_RIGHTSHIFT || (s) == KEY_LEFTCTRL || (s) == KEY_RIGHTCTRL || (s) == KEY_LEFTMETA || (s) == KEY_RIGHTMETA || (s) == KEY_LEFTALT || (s) == KEY_RIGHTALT || (s) == KEY_FN)

Functions

- int os_inputopen (usbdevice *kb)
- void os_inputclose (usbdevice *kb)
- void inputupdate (usbdevice *kb)
- void updateindicators_kb (usbdevice *kb, int force)
- void initbind (binding *bind)
- void freebind (binding *bind)
- void cmd_bind (usbdevice *kb, usbmode *mode, int dummy, int keyindex, const char *to)
- void cmd_unbind (usbdevice *kb, usbmode *mode, int dummy, int keyindex, const char *ignored)
- void cmd_rebind (usbdevice *kb, usbmode *mode, int dummy, int keyindex, const char *ignored)
- void cmd_macro (usbdevice *kb, usbmode *mode, const int notifynumber, const char *keys, const char *assignment)
- void os_keypress (usbdevice *kb, int scancode, int down)
- void os_mousemove (usbdevice *kb, int x, int y)
- int os_setupindicators (usbdevice *kb)

8.21.1 Macro Definition Documentation

8.21.1.1 #define IS_MOD(*s*) ((s) == KEY_CAPSLOCK || (s) == KEY_NUMLOCK || (s) == KEY_SCROLLLOCK || (s) == KEY_LEFTSHIFT || (s) == KEY_RIGHTSHIFT || (s) == KEY_LEFTCTRL || (s) == KEY_RIGHTCTRL || (s) == KEY_LEFTMETA || (s) == KEY_RIGHTMETA || (s) == KEY_LEFTALT || (s) == KEY_RIGHTALT || (s) == KEY_FN)

Definition at line 34 of file input.h.

Referenced by inputupdate_keys().

8.21.2 Function Documentation

8.21.2.1 void cmd_bind (usbdevice * *kb*, usbmode * *mode*, int *dummy*, int *keyindex*, const char * *to*)

Definition at line 188 of file input.c.

References binding::base, usbmode::bind, imutex, keymap, N_KEYS_INPUT, and key::scan.

```
188                                                                                 {
189     if(keyindex >= N_KEYS_INPUT)
190         return;
191     // Find the key to bind to
192     int tocode = 0;
193     if(sscanf(to, "#x%ux", &tocode) != 1 && sscanf(to, "#%u", &tocode) == 1 && tocode <
    N_KEYS_INPUT){
194         pthread_mutex_lock(imutex(kb));
195         mode->bind.base[keyindex] = tocode;
196         pthread_mutex_unlock(imutex(kb));
197         return;
198     }
199     // If not numeric, look it up
200     for(int i = 0; i < N_KEYS_INPUT; i++){
201         if(keymap[i].name && !strcmp(to, keymap[i].name)){
202             pthread_mutex_lock(imutex(kb));
203             mode->bind.base[keyindex] = keymap[i].scan;
204             pthread_mutex_unlock(imutex(kb));
205             return;
206         }
207     }
208 }
```

8.21.2.2 void cmd_macro (usbdevice * *kb*, usbmode * *mode*, const int *notifynumber*, const char * *keys*, const char * *assignment*)

Definition at line 332 of file input.c.

References _cmd_macro(), and imutex.

```
332
         {
333     pthread_mutex_lock(imutex(kb));
334     _cmd_macro(mode, keys, assignment);
335     pthread_mutex_unlock(imutex(kb));
336 }
```

Here is the call graph for this function:

8.21.2.3 void cmd_rebind (usbdevice * *kb*, usbmode * *mode*, int *dummy*, int *keyindex*, const char * *ignored*)

Definition at line 218 of file input.c.

References binding::base, usbmode::bind, imutex, keymap, N_KEYS_INPUT, and key::scan.

```
218                                                                                 {
219     if(keyindex >= N_KEYS_INPUT)
```

```
220         return;
221     pthread_mutex_lock(imutex(kb));
222     mode->bind.base[keyindex] = keymap[keyindex].scan;
223     pthread_mutex_unlock(imutex(kb));
224 }
```

8.21.2.4 void cmd_unbind (usbdevice * *kb*, usbmode * *mode*, int *dummy*, int *keyindex*, const char * *ignored*)

Definition at line 210 of file input.c.

References binding::base, usbmode::bind, imutex, KEY_UNBOUND, and N_KEYS_INPUT.

```
210                                                                                 {
211     if(keyindex >= N_KEYS_INPUT)
212         return;
213     pthread_mutex_lock(imutex(kb));
214     mode->bind.base[keyindex] = KEY_UNBOUND;
215     pthread_mutex_unlock(imutex(kb));
216 }
```

8.21.2.5 void freebind (binding * *bind*)

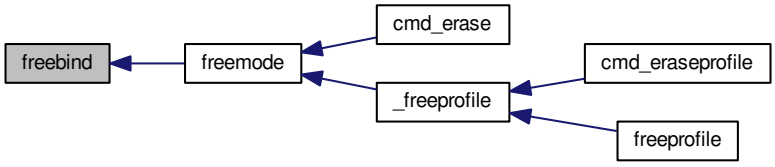
Definition at line 181 of file input.c.

References keymacro::actions, binding::macrocount, and binding::macros.

Referenced by freemode().

```
181                             {
182     for(int i = 0; i < bind->macrocount; i++)
183         free(bind->macros[i].actions);
184     free(bind->macros);
185     memset(bind, 0, sizeof(*bind));
186 }
```

Here is the caller graph for this function:



8.21.2.6 void initbind (binding * *bind*)

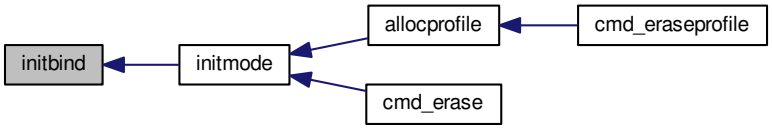
Definition at line 173 of file input.c.

References binding::base, keymap, binding::macrocap, binding::macrocount, binding::macros, N_KEYS_INPUT, and key::scan.

Referenced by initmode().

```
173                             {
174     for(int i = 0; i < N_KEYS_INPUT; i++)
175         bind->base[i] = keymap[i].scan;
176     bind->macros = calloc(32, sizeof(keymacro));
177     bind->macrocap = 32;
178     bind->macrocount = 0;
179 }
```

Here is the caller graph for this function:



8.21.2.7 void inputupdate (usbdevice * kb)

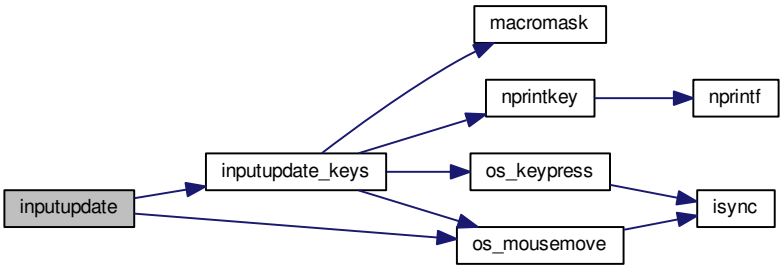
Definition at line 122 of file input.c.

References usbdevice::input, inputupdate_keys(), os_mousemove(), usbdevice::profile, usbinput::rel_x, usbinput::rel_y, usbdevice::uinput_kb, and usbdevice::uinput_mouse.

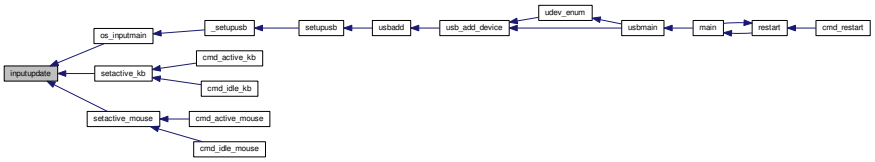
Referenced by os_inputmain(), setactive_kb(), and setactive_mouse().

```
122                                {
123 #ifdef OS_LINUX
124     if((!kb->uinput_kb || !kb->uinput_mouse)
125 #else
126     if(!kb->event
127 #endif
128             || !kb->profile)
129         return;
130     // Process key/button input
131     inputupdate_keys(kb);
132     // Process mouse movement
133     usbinput* input = &kb->input;
134     if(input->rel_x != 0 || input->rel_y != 0){
135         os_mousemove(kb, input->rel_x, input->rel_y);
136         input->rel_x = input->rel_y = 0;
137     }
138     // Finish up
139     memcpy(input->prevkeys, input->keys, N_KEYBYTES_INPUT);
140 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



8.21.2.8 void os_inputclose (usbdevice * *kb*)

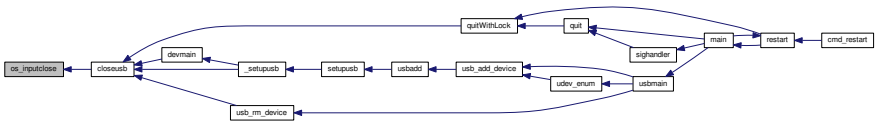
Definition at line 70 of file input_linux.c.

References ckb_warn, usbdevice::uinput_kb, and usbdevice::uinput_mouse.

Referenced by closeusb().

```
70                                  {
71     if(kb->uinput_kb <= 0 || kb->uinput_mouse <= 0)
72         return;
73     // Set all keys released
74     struct input_event event;
75     memset(&event, 0, sizeof(event));
76     event.type = EV_KEY;
77     for(int key = 0; key < KEY_CNT; key++){
78         event.code = key;
79         if(write(kb->uinput_kb - 1, &event, sizeof(event)) <= 0)
80             ckb_warn("uinput write failed: %s\n", strerror(errno));
81         if(write(kb->uinput_mouse - 1, &event, sizeof(event)) <= 0)
82             ckb_warn("uinput write failed: %s\n", strerror(errno));
83     }
84     event.type = EV_SYN;
85     event.code = SYN_REPORT;
86     if(write(kb->uinput_kb - 1, &event, sizeof(event)) <= 0)
87         ckb_warn("uinput write failed: %s\n", strerror(errno));
88     if(write(kb->uinput_mouse - 1, &event, sizeof(event)) <= 0)
89         ckb_warn("uinput write failed: %s\n", strerror(errno));
90     // Close the keyboard
91     ioctl(kb->uinput_kb - 1, UI_DEV_DESTROY);
92     close(kb->uinput_kb - 1);
93     kb->uinput_kb = 0;
94     // Close the mouse
95     ioctl(kb->uinput_mouse - 1, UI_DEV_DESTROY);
96     close(kb->uinput_mouse - 1);
97     kb->uinput_mouse = 0;
98 }
```

Here is the caller graph for this function:



8.21.2.9 int os_inputopen (usbdevice * *kb*)

Definition at line 49 of file input_linux.c.

References usbdevice::fwversion, INDEX_OF, keyboard, usbdevice::name, usbdevice::product, usbdevice::uinput_kb, usbdevice::uinput_mouse, uinputopen(), and usbdevice::vendor.

Referenced by _setupusb().

```
49                                {
50      // Create the new input device
51      int index = INDEX_OF(kb, keyboard);
52      struct uinput_user_dev indev;
53      memset(&indev, 0, sizeof(indev));
54      snprintf(indev.name, UINPUT_MAX_NAME_SIZE, "ckb%d: %s", index, kb->name);
55      indev.id.bustype = BUS_USB;
56      indev.id.vendor = kb->vendor;
57      indev.id.product = kb->product;
58      indev.id.version = kb->fwversion;
59      // Open keyboard
60      int fd = uinputopen(&indev, 0);
61      kb->uinput_kb = fd;
62      if(fd <= 0)
63          return 0;
64      // Open mouse
65      fd = uinputopen(&indev, 1);
66      kb->uinput_mouse = fd;
67      return fd <= 0;
68  }
```

Here is the call graph for this function:

Here is the caller graph for this function:

8.21.2.10 void os_keypress (usbdevice * *kb*, int *scancode*, int *down*)

Definition at line 112 of file input_linux.c.

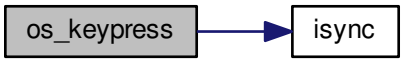
References BTN_WHEELDOWN, BTN_WHEELUP, ckb_warn, isync(), SCAN_MOUSE, usbdevice::uinput_kb, and usbdevice::uinput_mouse.

Referenced by inputupdate_keys().

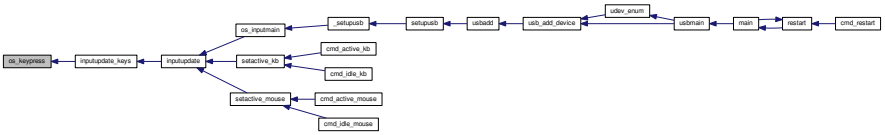
```
112                                                     {
113     struct input_event event;
114     memset(&event, 0, sizeof(event));
115     int is_mouse = 0;
116     if(scancode == BTN_WHEELUP || scancode == BTN_WHEELDOWN){
117         // The mouse wheel is a relative axis
118         if(!down)
119             return;
120         event.type = EV_REL;
121         event.code = REL_WHEEL;
122         event.value = (scancode == BTN_WHEELUP ? 1 : -1);
123         is_mouse = 1;
124     } else {
125         // Mouse buttons and key events are both EV_KEY. The scancodes are already correct, just remove the
    ckb bit
126         event.type = EV_KEY;
127         event.code = scancode & ~SCAN_MOUSE;
128         event.value = down;
129         is_mouse = !!(scancode & SCAN_MOUSE);
130     }
```

```
131     if(write((is_mouse ? kb->uinput_mouse : kb->uinput_kb) - 1, &event, sizeof(event))
        <= 0)
132         ckb_warn("uinput write failed: %s\n", strerror(errno));
133     else
134         isync(kb);
135 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



8.21.2.11 void os_mousemove (usbdevice * *kb*, int *x*, int *y*)

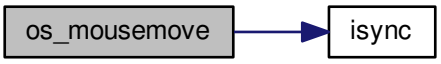
Definition at line 137 of file input_linux.c.

References ckb_warn, isync(), and usbdevice::uinput_mouse.

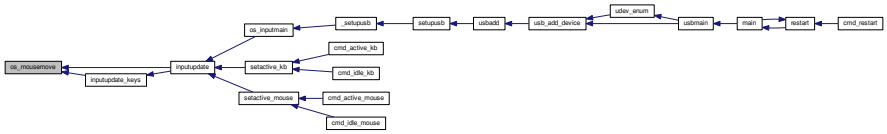
Referenced by inputupdate(), and inputupdate_keys().

```
137                                                 {
138     struct input_event event;
139     memset(&event, 0, sizeof(event));
140     event.type = EV_REL;
141     if(x != 0){
142         event.code = REL_X;
143         event.value = x;
144         if(write(kb->uinput_mouse - 1, &event, sizeof(event)) <= 0)
145             ckb_warn("uinput write failed: %s\n", strerror(errno));
146         else
147             isync(kb);
148     }
149     if(y != 0){
150         event.code = REL_Y;
151         event.value = y;
152         if(write(kb->uinput_mouse - 1, &event, sizeof(event)) <= 0)
153             ckb_warn("uinput write failed: %s\n", strerror(errno));
154         else
155             isync(kb);
156     }
157 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



8.21.2.12 int os_setupindicators (usbdevice * *kb*)

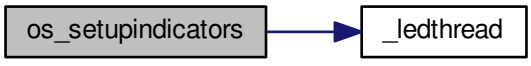
Definition at line 183 of file input_linux.c.

References _ledthread(), usbdevice::hw_ileds, usbdevice::hw_ileds_old, and usbdevice::ileds.

Referenced by _setupusb().

```
183                                     {
184     // Initialize LEDs to all off
185     kb->hw_ileds = kb->hw_ileds_old = kb->ileds = 0;
186     // Create and detach thread to read LED events
187     pthread_t thread;
188     int err = pthread_create(&thread, 0, _ledthread, kb);
189     if(err != 0)
190         return err;
191     pthread_detach(thread);
192     return 0;
193 }
```

Here is the call graph for this function:



Here is the caller graph for this function:

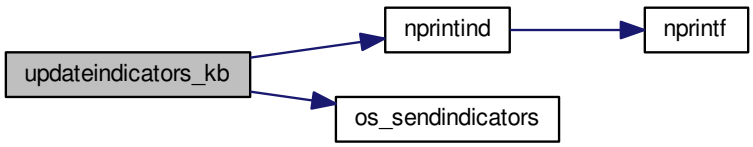
8.21.2.13 void updateindicators_kb (usbdevice * *kb*, int *force*)

Definition at line 142 of file input.c.

References usbdevice::active, usbprofile::currentmode, DELAY_SHORT, usbdevice::hw_ileds, usbdevice::hw_ileds_old, I_CAPS, I_NUM, I_SCROLL, usbdevice::ileds, usbmode::inotify, usbmode::ioff, usbmode::ion, nprintind(), os_sendindicators(), OUTFIFO_MAX, and usbdevice::profile.

```
142                                                 {
143     // Read current hardware indicator state (set externally)
144     uchar old = kb->ileds, hw_old = kb->hw_ileds_old;
145     uchar new = kb->hw_ileds, hw_new = new;
146     // Update them if needed
147     if(kb->active){
148         usbmode* mode = kb->profile->currentmode;
149         new = (new & ~mode->ioff) | mode->ion;
150     }
151     kb->ileds = new;
152     kb->hw_ileds_old = hw_new;
153     if(old != new || force){
154         DELAY_SHORT(kb);
155         os_sendindicators(kb);
156     }
157     // Print notifications if desired
158     if(!kb->active)
159         return;
160     usbmode* mode = kb->profile->currentmode;
161     uchar indicators[] = { I_NUM, I_CAPS, I_SCROLL };
162     for(unsigned i = 0; i < sizeof(indicators) / sizeof(uchar); i++){
163         uchar mask = indicators[i];
164         if((hw_old & mask) == (hw_new & mask))
165             continue;
166         for(int notify = 0; notify < OUTFIFO_MAX; notify++){
167             if(mode->inotify[notify] & mask)
168                 nprintind(kb, notify, mask, hw_new & mask);
169         }
170     }
171 }
```

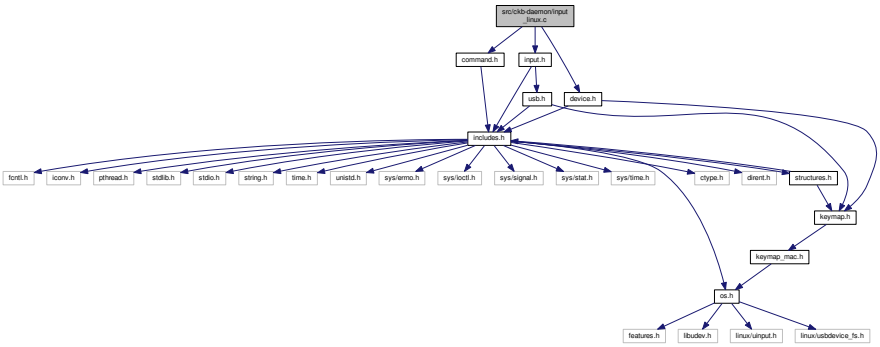
Here is the call graph for this function:



8.22 src/ckb-daemon/input_linux.c File Reference

```
#include "command.h"
#include "device.h"
#include "input.h"
```

Include dependency graph for input_linux.c:



Functions

- int uinputopen (struct uinput_user_dev *indev, int mouse)
- int os_inputopen (usbdevice *kb)
- void os_inputclose (usbdevice *kb)
- static void isync (usbdevice *kb)
- void os_keypress (usbdevice *kb, int scancode, int down)
- void os_mousemove (usbdevice *kb, int x, int y)
- void * _ledthread (void *ctx)
- int os_setupindicators (usbdevice *kb)

8.22.1 Function Documentation

8.22.1.1 void* _ledthread (void * *ctx*)

Definition at line 159 of file input_linux.c.

References dmutex, usbdevice::hw_ileds, usbdevice::uinput_kb, and usbdevice::vtable.

Referenced by os_setupindicators().

```
159                          {
160     usbdevice* kb = ctx;
161     uchar ileds = 0;
162     // Read LED events from the uinput device
163     struct input_event event;
164     while(read(kb->uinput_kb - 1, &event, sizeof(event)) > 0){
165         if(event.type == EV_LED && event.code < 8){
166             char which = 1 << event.code;
167             if(event.value)
168                 ileds |= which;
169             else
170                 ileds &= ~which;
171         }
172         // Update them if needed
173         pthread_mutex_lock(dmutex(kb));
174         if(kb->hw_ileds != ileds){
175             kb->hw_ileds = ileds;
176             kb->vtable->updateindicators(kb, 0);
177         }
178         pthread_mutex_unlock(dmutex(kb));
179     }
180     return 0;
181 }
```

Here is the caller graph for this function:

8.22.1.2 static void isync (usbdevice * kb) [static]

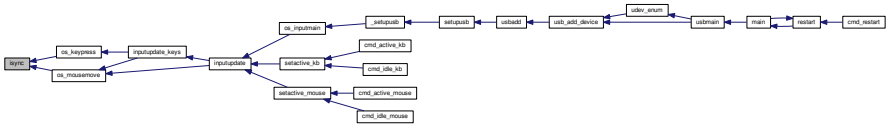
Definition at line 101 of file input_linux.c.

References ckb_warn, usbdevice::uinput_kb, and usbdevice::uinput_mouse.

Referenced by os_keypress(), and os_mousemove().

```
101                                 {
102     struct input_event event;
103     memset(&event, 0, sizeof(event));
104     event.type = EV_SYN;
105     event.code = SYN_REPORT;
106     if(write(kb->uinput_kb - 1, &event, sizeof(event)) <= 0)
107         ckb_warn("uinput write failed: %s\n", strerror(errno));
108     if(write(kb->uinput_mouse - 1, &event, sizeof(event)) <= 0)
109         ckb_warn("uinput write failed: %s\n", strerror(errno));
110 }
```

Here is the caller graph for this function:



8.22.1.3 void os_inputclose (usbdevice * kb)

Definition at line 70 of file input_linux.c.

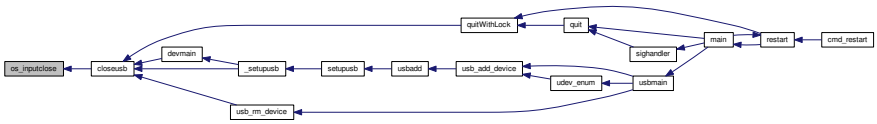
References ckb_warn, usbdevice::uinput_kb, and usbdevice::uinput_mouse.

Referenced by closeusb().

```
70                                  {
71     if(kb->uinput_kb <= 0 || kb->uinput_mouse <= 0)
72         return;
73     // Set all keys released
74     struct input_event event;
75     memset(&event, 0, sizeof(event));
76     event.type = EV_KEY;
77     for(int key = 0; key < KEY_CNT; key++){
78         event.code = key;
79         if(write(kb->uinput_kb - 1, &event, sizeof(event)) <= 0)
80             ckb_warn("uinput write failed: %s\n", strerror(errno));
81         if(write(kb->uinput_mouse - 1, &event, sizeof(event)) <= 0)
82             ckb_warn("uinput write failed: %s\n", strerror(errno));
83     }
84     event.type = EV_SYN;
85     event.code = SYN_REPORT;
86     if(write(kb->uinput_kb - 1, &event, sizeof(event)) <= 0)
87         ckb_warn("uinput write failed: %s\n", strerror(errno));
88     if(write(kb->uinput_mouse - 1, &event, sizeof(event)) <= 0)
89         ckb_warn("uinput write failed: %s\n", strerror(errno));
90     // Close the keyboard
91     ioctl(kb->uinput_kb - 1, UI_DEV_DESTROY);
92     close(kb->uinput_kb - 1);
93     kb->uinput_kb = 0;
```

```
94      // Close the mouse
95      ioctl(kb->uinput_mouse - 1, UI_DEV_DESTROY);
96      close(kb->uinput_mouse - 1);
97      kb->uinput_mouse = 0;
98  }
```

Here is the caller graph for this function:



8.22.1.4 int os_inputopen (usbdevice * *kb*)

Definition at line 49 of file input_linux.c.

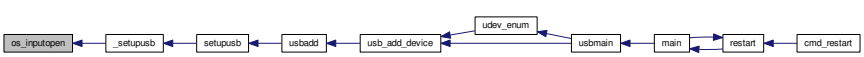
References usbdevice::fwversion, INDEX_OF, keyboard, usbdevice::name, usbdevice::product, usbdevice::uinput-_kb, usbdevice::uinput_mouse, uinputopen(), and usbdevice::vendor.

Referenced by _setupusb().

```
49                                {
50      // Create the new input device
51      int index = INDEX_OF(kb, keyboard);
52      struct uinput_user_dev indev;
53      memset(&indev, 0, sizeof(indev));
54      snprintf(indev.name, UINPUT_MAX_NAME_SIZE, "ckb%d: %s", index, kb->name);
55      indev.id.bustype = BUS_USB;
56      indev.id.vendor = kb->vendor;
57      indev.id.product = kb->product;
58      indev.id.version = kb->fwversion;
59      // Open keyboard
60      int fd = uinputopen(&indev, 0);
61      kb->uinput_kb = fd;
62      if(fd <= 0)
63          return 0;
64      // Open mouse
65      fd = uinputopen(&indev, 1);
66      kb->uinput_mouse = fd;
67      return fd <= 0;
68  }
```

Here is the call graph for this function:

Here is the caller graph for this function:



8.22.1.5 void os_keypress (usbdevice * *kb*, int *scancode*, int *down*)

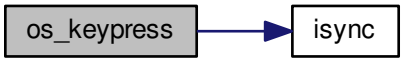
Definition at line 112 of file input_linux.c.

References BTN_WHEELDOWN, BTN_WHEELUP, ckb_warn, isync(), SCAN_MOUSE, usbdevice::uinput_kb, and usbdevice::uinput_mouse.

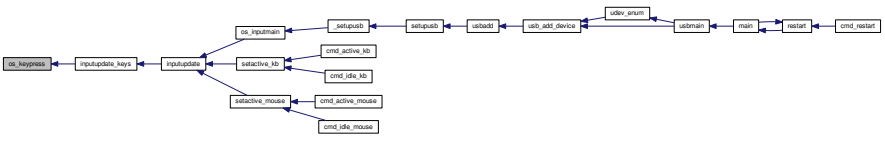
Referenced by inputupdate_keys().

```
112                                                     {
113     struct input_event event;
114     memset(&event, 0, sizeof(event));
115     int is_mouse = 0;
116     if(scancode == BTN_WHEELUP || scancode == BTN_WHEELDOWN){
117         // The mouse wheel is a relative axis
118         if(!down)
119             return;
120         event.type = EV_REL;
121         event.code = REL_WHEEL;
122         event.value = (scancode == BTN_WHEELUP ? 1 : -1);
123         is_mouse = 1;
124     } else {
125         // Mouse buttons and key events are both EV_KEY. The scancodes are already correct, just remove the
      ckb bit
126         event.type = EV_KEY;
127         event.code = scancode & ~SCAN_MOUSE;
128         event.value = down;
129         is_mouse = !!(scancode & SCAN_MOUSE);
130     }
131     if(write((is_mouse ? kb->uinput_mouse : kb->uinput_kb) - 1, &event, sizeof(event))
      <= 0)
132         ckb_warn("uinput write failed: %s\n", strerror(errno));
133     else
134         isync(kb);
135 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



8.22.1.6 void os_mousemove (usbdevice * *kb*, int *x*, int *y*)

Definition at line 137 of file input_linux.c.

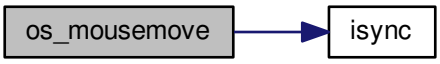
References ckb_warn, isync(), and usbdevice::uinput_mouse.

Referenced by inputupdate(), and inputupdate_keys().

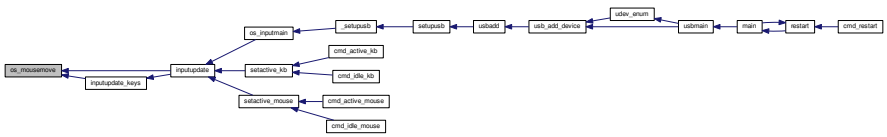
```
137                                                 {
```

```
138     struct input_event event;
139     memset(&event, 0, sizeof(event));
140     event.type = EV_REL;
141     if(x != 0){
142         event.code = REL_X;
143         event.value = x;
144         if(write(kb->uinput_mouse - 1, &event, sizeof(event)) <= 0)
145             ckb_warn("uinput write failed: %s\n", strerror(errno));
146         else
147             isync(kb);
148     }
149     if(y != 0){
150         event.code = REL_Y;
151         event.value = y;
152         if(write(kb->uinput_mouse - 1, &event, sizeof(event)) <= 0)
153             ckb_warn("uinput write failed: %s\n", strerror(errno));
154         else
155             isync(kb);
156     }
157 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



8.22.1.7 int os_setupindicators (usbdevice * *kb*)

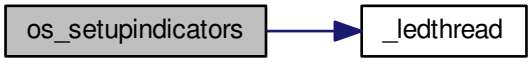
Definition at line 183 of file input_linux.c.

References _ledthread(), usbdevice::hw_ileds, usbdevice::hw_ileds_old, and usbdevice::ileds.

Referenced by _setupusb().

```
183                                     {
184     // Initialize LEDs to all off
185     kb->hw_ileds = kb->hw_ileds_old = kb->ileds = 0;
186     // Create and detach thread to read LED events
187     pthread_t thread;
188     int err = pthread_create(&thread, 0, _ledthread, kb);
189     if(err != 0)
190         return err;
191     pthread_detach(thread);
192     return 0;
193 }
```

Here is the call graph for this function:



Here is the caller graph for this function:

8.22.1.8 int uinputopen (struct uinput_user_dev * *indev*, int *mouse*)

Definition at line 9 of file input_linux.c.

References ckb_err, and ckb_warn.

Referenced by os_inputopen().

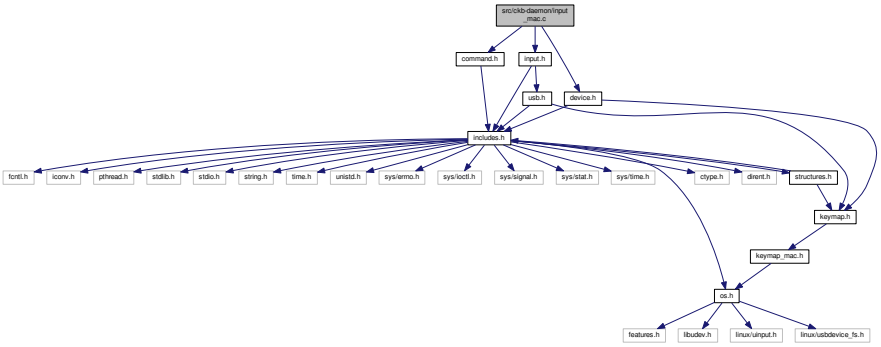
```
9                                                                   {
10      int fd = open("/dev/uinput", O_RDWR);
11      if(fd < 0){
12          // If that didn't work, try /dev/input/uinput instead
13          fd = open("/dev/input/uinput", O_RDWR);
14          if(fd < 0){
15              ckb_err("Failed to open uinput: %s\n", strerror(errno));
16              return 0;
17          }
18      }
19      // Enable all keys and mouse buttons
20      ioctl(fd, UI_SET_EVBIT, EV_KEY);
21      for(int i = 0; i < KEY_CNT; i++)
22          ioctl(fd, UI_SET_KEYBIT, i);
23      if(mouse){
24          // Enable mouse axes
25          ioctl(fd, UI_SET_EVBIT, EV_REL);
26          for(int i = 0; i < REL_CNT; i++)
27              ioctl(fd, UI_SET_RELBIT, i);
28      } else {
29          // Enable LEDs
30          ioctl(fd, UI_SET_EVBIT, EV_LED);
31          for(int i = 0; i < LED_CNT; i++)
32              ioctl(fd, UI_SET_LEDBIT, i);
33          // Eanble autorepeat
34          ioctl(fd, UI_SET_EVBIT, EV_REP);
35      }
36      // Enable sychronization
37      ioctl(fd, UI_SET_EVBIT, EV_SYN);
38      // Create the device
39      if(write(fd, indev, sizeof(*indev)) <= 0)
40          ckb_warn("uinput write failed: %s\n", strerror(errno));
41      if(ioctl(fd, UI_DEV_CREATE)){
42          ckb_err("Failed to create uinput device: %s\n", strerror(errno));
43          close(fd);
44          return 0;
45      }
46      return fd + 1;
47  }
```

Here is the caller graph for this function:

8.23 src/ckb-daemon/input_mac.c File Reference

```
#include "command.h"
#include "device.h"
#include "input.h"
```

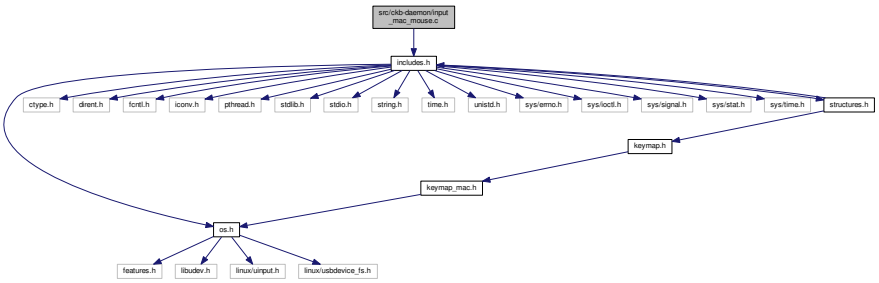
Include dependency graph for input_mac.c:



8.24 src/ckb-daemon/input_mac_mouse.c File Reference

```
#include "includes.h"
```

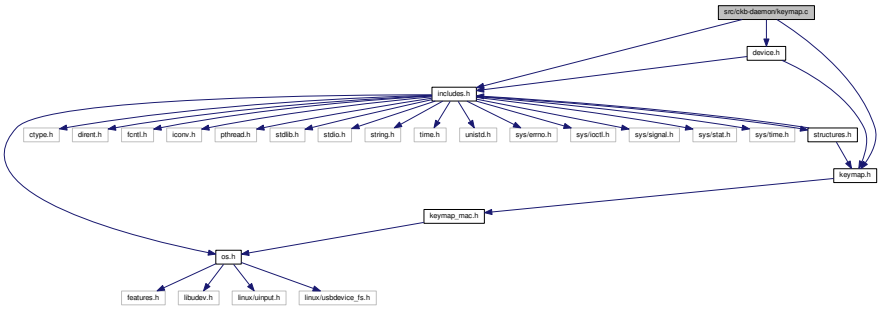
Include dependency graph for input_mac_mouse.c:



8.25 src/ckb-daemon/keymap.c File Reference

```
#include "device.h"
#include "includes.h"
#include "keymap.h"
```

Include dependency graph for keymap.c:



Macros

- #define BUTTON_HID_COUNT 5

Functions

- void hid_kb_translate (unsigned char ∗kbinput, int endpoint, int length, const unsigned char ∗urbinput)
- void hid_mouse_translate (unsigned char ∗kbinput, short ∗xaxis, short ∗yaxis, int endpoint, int length, const unsigned char ∗urbinput)
- void corsair_kbcopy (unsigned char ∗kbinput, int endpoint, const unsigned char ∗urbinput)
- void corsair_mousecopy (unsigned char ∗kbinput, int endpoint, const unsigned char ∗urbinput)

Variables

- const key keymap [(((152+3+12)+25)+11)]

8.25.1 Macro Definition Documentation

8.25.1.1 #define BUTTON_HID_COUNT 5

Definition at line 364 of file keymap.c.

Referenced by corsair_mousecopy(), and hid_mouse_translate().

8.25.2 Function Documentation

8.25.2.1 void corsair_kbcopy (unsigned char ∗ *kbinput*, int *endpoint*, const unsigned char ∗ *urbinput*)

Definition at line 394 of file keymap.c.

References N_KEYBYTES_HW.

Referenced by os_inputmain().

```
394                                                                                 {
395     if(endpoint == 2 || endpoint == -2){
396         if(urbinput[0] != 3)
397             return;
398         urbinput++;
399     }
400     memcpy(kbinput, urbinput, N_KEYBYTES_HW);
401 }
```

Here is the caller graph for this function:

8.25.2.2 void corsair_mousecopy (unsigned char * *kbinput*, int *endpoint*, const unsigned char * *urbinput*)

Definition at line 403 of file keymap.c.

References BUTTON_HID_COUNT, CLEAR_KEYBIT, MOUSE_BUTTON_FIRST, N_BUTTONS_HW, and SET_KEYBIT.

Referenced by os_inputmain().

```
403                                                                                                                 {
404     if(endpoint == 2 || endpoint == -2){
405         if(urbinput[0] != 3)
406             return;
407         urbinput++;
408     }
409     for(int bit = BUTTON_HID_COUNT; bit < N_BUTTONS_HW; bit++){
410         int byte = bit / 8;
411         uchar test = 1 << (bit % 8);
412         if(urbinput[byte] & test)
413             SET_KEYBIT(kbinput, MOUSE_BUTTON_FIRST + bit);
414         else
415             CLEAR_KEYBIT(kbinput, MOUSE_BUTTON_FIRST + bit);
416     }
417 }
```

Here is the caller graph for this function:

8.25.2.3 void hid_kb_translate (unsigned char * *kbinput*, int *endpoint*, int *length*, const unsigned char * *urbinput*)

Definition at line 223 of file keymap.c.

References ckb_warn, CLEAR_KEYBIT, and SET_KEYBIT.

Referenced by os_inputmain().

```
223                                                                                                                 {
224     if(length < 1)
225         return;
226     // LUT for HID -> Corsair scancodes (-1 for no scan code, -2 for currently unsupported)
227     // Modified from Linux drivers/hid/usbhid/usbkbd.c, key codes replaced with array indices and K95 keys
        added
228     static const short hid_codes[256] = {
229         -1,  -1,  -1,  -1,  37,  54,  52,  39,  27,  40,  41,  42,  32,  43,  44,  45,
230         56,  55,  33,  34,  25,  28,  38,  29,  31,  53,  26,  51,  30,  50,  13,  14,
231         15,  16,  17,  18,  19,  20,  21,  22,  82,   0,  86,  24,  64,  23,  84,  35,
232         79,  80,  81,  46,  47,  12,  57,  58,  59,  36,   1,   2,   3,   4,   5,   6,
233          7,   8,   9,  10,  11,  72,  73,  74,  75,  76,  77,  78,  87,  88,  89,  95,
234         93,  94,  92, 102, 103, 104, 105, 106, 107, 115, 116, 117, 112, 113, 114, 108,
235        109, 110, 118, 119,  49,  69,  -2,  -2,  -2,  -2,  -2,  -2,  -2,  -2,  -2,  -2,
236         -2,  -2,  -2,  -2,  -2,  -2,  -2,  -2,  98,  -2,  -2,  -2,  -2,  -2,  -2,  97,
237        130, 131,  -1,  -1,  -1,  -2,  -1,  -2,  -2,  -2,  -2,  -2,  -2,  -1,  -1,  -1,
238         -2,  -2,  -2,  -2,  -2,  -1,  -1,  -1,  -1,  -1,  -1,  -1,  -1,  -1,  -1,  -1,
239         -1,  -1,  -1,  -1,  -1,  -1,  -1,  -1,  -1,  -1,  -1,  -1,  -1,  -1,  -1,  -1,
240         -1,  -1,  -1,  -1,  -1,  -1,  -1,  -1,  -1,  -1,  -1,  -1,  -1,  -1,  -1,  -1,
241         -1,  -1,  -1,  -1,  -1,  -1,  -1,  -1,  -1,  -1,  -1,  -1,  -3,  -1,  -1,  -1,  // <- -3 = non-RGB
        program key
```

```
242         120, 121, 122, 123, 124, 125, 126, 127, 128, 129, 136, 137, 138, 139, 140, 141,
243          60,  48,  62,  61,  91,  90,  67,  68, 142, 143,  99, 101,  -2, 130, 131,  97,
244          -2, 133, 134, 135,  -2,  96,  -2, 132,  -2,  -2,  71,  71,  71,  71,  -1,  -1,
245     };
246     switch(endpoint){
247     case 1:
248     case -1:
249         // EP 1: 6KRO input (RGB and non-RGB)
250         // Clear previous input
251         for(int i = 0; i < 256; i++){
252             if(hid_codes[i] >= 0)
253                 CLEAR_KEYBIT(kbinput, hid_codes[i]);
254         }
255         // Set new input
256         for(int i = 0; i < 8; i++){
257             if((urbinput[0] >> i) & 1)
258                 SET_KEYBIT(kbinput, hid_codes[i + 224]);
259         }
260         for(int i = 2; i < length; i++){
261             if(urbinput[i] > 3){
262                 int scan = hid_codes[urbinput[i]];
263                 if(scan >= 0)
264                     SET_KEYBIT(kbinput, scan);
265                 else
266                     ckb_warn("Got unknown key press %d on EP 1\n", urbinput[i]);
267             }
268         }
269         break;
270     case -2:
271         // EP 2 RGB: NKRO input
272         if(urbinput[0] == 1){
273             // Type 1: standard key
274             if(length != 21)
275                 return;
276             for(int bit = 0; bit < 8; bit++){
277                 if((urbinput[1] >> bit) & 1)
278                     SET_KEYBIT(kbinput, hid_codes[bit + 224]);
279                 else
280                     CLEAR_KEYBIT(kbinput, hid_codes[bit + 224]);
281             }
282             for(int byte = 0; byte < 19; byte++){
283                 char input = urbinput[byte + 2];
284                 for(int bit = 0; bit < 8; bit++){
285                     int keybit = byte * 8 + bit;
286                     int scan = hid_codes[keybit];
287                     if((input >> bit) & 1){
288                         if(scan >= 0)
289                             SET_KEYBIT(kbinput, hid_codes[keybit]);
290                         else
291                             ckb_warn("Got unknown key press %d on EP 2\n", keybit);
292                     } else if(scan >= 0)
293                         CLEAR_KEYBIT(kbinput, hid_codes[keybit]);
294                 }
295             }
296             break;
297         } else if(urbinput[0] == 2)
298             ;       // Type 2: media key (fall through)
299         else
300             break;  // No other known types
301     case 2:
302         // EP 2 Non-RGB: media keys
303         CLEAR_KEYBIT(kbinput, 97);          // mute
304         CLEAR_KEYBIT(kbinput, 98);          // stop
305         CLEAR_KEYBIT(kbinput, 99);          // prev
306         CLEAR_KEYBIT(kbinput, 100);         // play
307         CLEAR_KEYBIT(kbinput, 101);         // next
308         CLEAR_KEYBIT(kbinput, 130);         // volup
309         CLEAR_KEYBIT(kbinput, 131);         // voldn
310         for(int i = 0; i < length; i++){
311             switch(urbinput[i]){
312             case 181:
313                 SET_KEYBIT(kbinput, 101);   // next
314                 break;
315             case 182:
316                 SET_KEYBIT(kbinput, 99);    // prev
317                 break;
318             case 183:
319                 SET_KEYBIT(kbinput, 98);    // stop
320                 break;
321             case 205:
322                 SET_KEYBIT(kbinput, 100);   // play
323                 break;
324             case 226:
325                 SET_KEYBIT(kbinput, 97);    // mute
326                 break;
327             case 233:
328                 SET_KEYBIT(kbinput, 130);   // volup
```

```
329                 break;
330             case 234:
331                 SET_KEYBIT(kbinput, 131);   // voldn
332                 break;
333             }
334         }
335         break;
336     case 3:
337         // EP 3 non-RGB: NKRO input
338         if(length != 15)
339             return;
340         for(int bit = 0; bit < 8; bit++){
341             if((urbinput[0] >> bit) & 1)
342                 SET_KEYBIT(kbinput, hid_codes[bit + 224]);
343             else
344                 CLEAR_KEYBIT(kbinput, hid_codes[bit + 224]);
345         }
346         for(int byte = 0; byte < 14; byte++){
347             char input = urbinput[byte + 1];
348             for(int bit = 0; bit < 8; bit++){
349                 int keybit = byte * 8 + bit;
350                 int scan = hid_codes[keybit];
351                 if((input >> bit) & 1){
352                     if(scan >= 0)
353                         SET_KEYBIT(kbinput, hid_codes[keybit]);
354                     else
355                         ckb_warn("Got unknown key press %d on EP 3\n", keybit);
356                 } else if(scan >= 0)
357                     CLEAR_KEYBIT(kbinput, hid_codes[keybit]);
358             }
359         }
360         break;
361     }
362 }
```

Here is the caller graph for this function:

8.25.2.4 void hid_mouse_translate (unsigned char * *kbinput*, short * *xaxis*, short * *yaxis*, int *endpoint*, int *length*, const unsigned char * *urbinput*)

Definition at line 366 of file keymap.c.

References BUTTON_HID_COUNT, CLEAR_KEYBIT, MOUSE_BUTTON_FIRST, MOUSE_EXTRA_FIRST, and SET_KEYBIT.

Referenced by os_inputmain().

```
366
                                {
367     if((endpoint != 2 && endpoint != -2) || length < 10)
368         return;
369     // EP 2: mouse input
370     if(urbinput[0] != 1)
371         return;
372     // Byte 1 = mouse buttons (bitfield)
373     for(int bit = 0; bit < BUTTON_HID_COUNT; bit++){
374         if(urbinput[1] & (1 << bit))
375             SET_KEYBIT(kbinput, MOUSE_BUTTON_FIRST + bit);
376         else
377             CLEAR_KEYBIT(kbinput, MOUSE_BUTTON_FIRST + bit);
378     }
379     // Bytes 5 - 8: movement
380     *xaxis += *(short*)(urbinput + 5);
381     *yaxis += *(short*)(urbinput + 7);
382     // Byte 9: wheel
383     char wheel = urbinput[9];
384     if(wheel > 0)
385         SET_KEYBIT(kbinput, MOUSE_EXTRA_FIRST);         // wheelup
386     else
387         CLEAR_KEYBIT(kbinput, MOUSE_EXTRA_FIRST);
388     if(wheel < 0)
```

```
389         SET_KEYBIT(kbinput, MOUSE_EXTRA_FIRST + 1);     // wheeldn
390     else
391         CLEAR_KEYBIT(kbinput, MOUSE_EXTRA_FIRST + 1);
392 }
```

Here is the caller graph for this function:

8.25.3 Variable Documentation

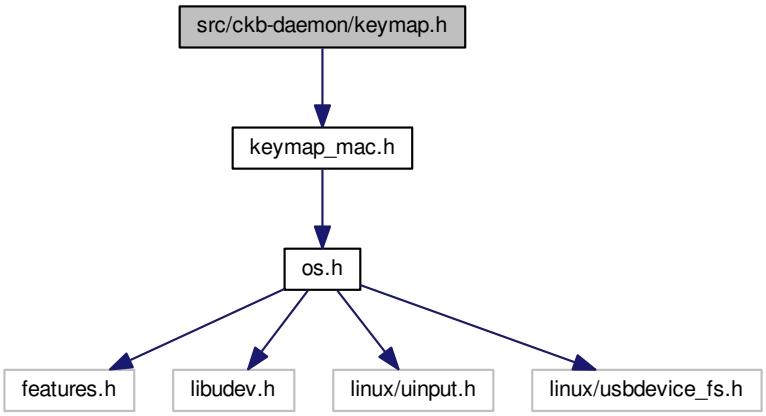
8.25.3.1 const key keymap[(((152+3+12)+25)+11)]

Definition at line 5 of file keymap.c.

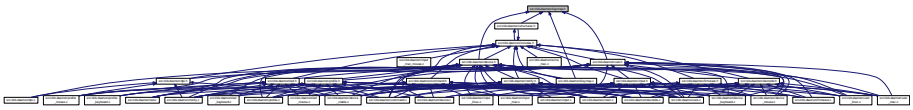
Referenced by _cmd_get(), _cmd_macro(), cmd_bind(), cmd_rebind(), cmd_rgb(), initbind(), inputupdate_keys(), nprintkey(), printrgb(), readcmd(), and setactive_kb().

8.26 src/ckb-daemon/keymap.h File Reference

`#include "keymap_mac.h"`
Include dependency graph for keymap.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct key

Macros

- #define KEY_NONE -1
- #define KEY_CORSAIR -2
- #define KEY_UNBOUND -3
- #define BTN_WHEELUP 0x1f01
- #define BTN_WHEELDOWN 0x1f02
- #define KEY_BACKSLASH_ISO KEY_BACKSLASH
- #define N_KEYS_HW 152
- #define N_KEYBYTES_HW ((N_KEYS_HW + 7) / 8)
- #define N_KEY_ZONES 3
- #define N_KEYS_EXTRA 12
- #define N_BUTTONS_HW 20
- #define N_BUTTONS_EXTENDED 25
- #define MOUSE_BUTTON_FIRST (N_KEYS_HW + N_KEY_ZONES + N_KEYS_EXTRA)
- #define MOUSE_EXTRA_FIRST (MOUSE_BUTTON_FIRST + N_BUTTONS_HW)
- #define N_KEYS_INPUT (MOUSE_BUTTON_FIRST + N_BUTTONS_EXTENDED)
- #define N_KEYBYTES_INPUT ((N_KEYS_INPUT + 7) / 8)
- #define LED_MOUSE N_KEYS_HW
- #define N_MOUSE_ZONES 5
- #define N_MOUSE_ZONES_EXTENDED 11
- #define LED_DPI (LED_MOUSE + 2)
- #define N_KEYS_EXTENDED (N_KEYS_INPUT + N_MOUSE_ZONES_EXTENDED)
- #define N_KEYBYTES_EXTENDED ((N_KEYS_EXTENDED + 7) / 8)
- #define SCAN_SILENT 0x8000
- #define SCAN_KBD 0
- #define SCAN_MOUSE 0x1000

Functions

- void hid_kb_translate (unsigned char *kbinput, int endpoint, int length, const unsigned char *urbinput)
- void hid_mouse_translate (unsigned char *kbinput, short *xaxis, short *yaxis, int endpoint, int length, const unsigned char *urbinput)
- void corsair_kbcopy (unsigned char *kbinput, int endpoint, const unsigned char *urbinput)
- void corsair_mousecopy (unsigned char *kbinput, int endpoint, const unsigned char *urbinput)

Variables

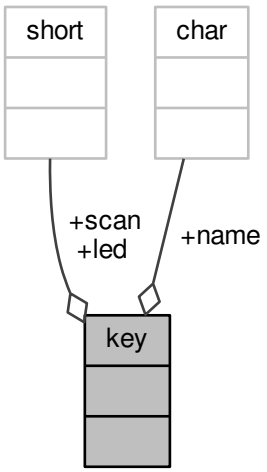
- const key keymap [(((152+3+12)+25)+11)]

8.26.1 Data Structure Documentation

8.26.1.1 struct key

Definition at line 49 of file keymap.h.

Collaboration diagram for key:



Data Fields

short	led	
const char *	name	
short	scan	

8.26.2 Macro Definition Documentation

8.26.2.1 #define BTN_WHEELDOWN 0x1f02

Definition at line 13 of file keymap.h.

Referenced by os_keypress().

8.26.2.2 #define BTN_WHEELUP 0x1f01

Definition at line 12 of file keymap.h.

Referenced by os_keypress().

8.26.2.3 #define KEY_BACKSLASH_ISO KEY_BACKSLASH

Definition at line 20 of file keymap.h.

8.26.2.4 #define KEY_CORSAIR -2

Definition at line 8 of file keymap.h.

8.26.2.5 #define KEY_NONE -1

Definition at line 7 of file keymap.h.

8.26.2.6 #define KEY_UNBOUND -3

Definition at line 9 of file keymap.h.

Referenced by cmd_unbind().

8.26.2.7 #define LED_DPI (LED_MOUSE + 2)

Definition at line 43 of file keymap.h.

Referenced by loadrgb_mouse(), and savergb_mouse().

8.26.2.8 #define LED_MOUSE N_KEYS_HW

Definition at line 39 of file keymap.h.

Referenced by isblack(), loaddpi(), loadrgb_mouse(), rgbcmp(), savedpi(), savergb_mouse(), and updatergb_mouse().

8.26.2.9 #define MOUSE_BUTTON_FIRST (N_KEYS_HW + N_KEY_ZONES + N_KEYS_EXTRA)

Definition at line 33 of file keymap.h.

Referenced by corsair_mousecopy(), and hid_mouse_translate().

8.26.2.10 #define MOUSE_EXTRA_FIRST (MOUSE_BUTTON_FIRST + N_BUTTONS_HW)

Definition at line 34 of file keymap.h.

Referenced by hid_mouse_translate().

8.26.2.11 #define N_BUTTONS_EXTENDED 25

Definition at line 32 of file keymap.h.

8.26.2.12 #define N_BUTTONS_HW 20

Definition at line 31 of file keymap.h.

Referenced by corsair_mousecopy().

8.26.2.13 #define N_KEY_ZONES 3

Definition at line 27 of file keymap.h.

8.26.2.14 #define N_KEYBYTES_EXTENDED ((N_KEYS_EXTENDED + 7) / 8)

Definition at line 46 of file keymap.h.

8.26.2.15 #define N_KEYBYTES_HW ((N_KEYS_HW + 7) / 8)

Definition at line 25 of file keymap.h.

Referenced by corsair_kbcopy().

8.26.2.16 #define N_KEYBYTES_INPUT ((N_KEYS_INPUT + 7) / 8)

Definition at line 37 of file keymap.h.

Referenced by _cmd_macro(), inputupdate_keys(), and macromask().

8.26.2.17 #define N_KEYS_EXTENDED (N_KEYS_INPUT + N_MOUSE_ZONES_EXTENDED)

Definition at line 45 of file keymap.h.

Referenced by printrgb(), and readcmd().

8.26.2.18 #define N_KEYS_EXTRA 12

Definition at line 29 of file keymap.h.

8.26.2.19 #define N_KEYS_HW 152

Definition at line 24 of file keymap.h.

Referenced by loadrgb_kb(), makergb_512(), rgbcmp(), and setactive_kb().

8.26.2.20 #define N_KEYS_INPUT (MOUSE_BUTTON_FIRST + N_BUTTONS_EXTENDED)

Definition at line 36 of file keymap.h.

Referenced by _cmd_get(), _cmd_macro(), cmd_bind(), cmd_notify(), cmd_rebind(), cmd_unbind(), initbind(), and inputupdate_keys().

8.26.2.21 #define N_MOUSE_ZONES 5

Definition at line 40 of file keymap.h.

Referenced by isblack(), loaddpi(), rgbcmp(), savedpi(), and updatergb_mouse().

8.26.2.22 #define N_MOUSE_ZONES_EXTENDED 11

Definition at line 41 of file keymap.h.

8.26.2.23 #define SCAN_KBD 0

Definition at line 57 of file keymap.h.

8.26.2.24 #define SCAN_MOUSE 0x1000

Definition at line 58 of file keymap.h.

Referenced by os_keypress().

8.26.2.25 #define SCAN_SILENT 0x8000

Definition at line 56 of file keymap.h.

Referenced by inputupdate_keys().

8.26.3 Function Documentation

8.26.3.1 void corsair_kbcopy (unsigned char * *kbinput*, int *endpoint*, const unsigned char * *urbinput*)

Definition at line 394 of file keymap.c.

References N_KEYBYTES_HW.

Referenced by os_inputmain().

```
394                                                                                                 {
395     if(endpoint == 2 || endpoint == -2){
396         if(urbinput[0] != 3)
397             return;
398         urbinput++;
399     }
400     memcpy(kbinput, urbinput, N_KEYBYTES_HW);
401 }
```

Here is the caller graph for this function:

8.26.3.2 void corsair_mousecopy (unsigned char * *kbinput*, int *endpoint*, const unsigned char * *urbinput*)

Definition at line 403 of file keymap.c.

References BUTTON_HID_COUNT, CLEAR_KEYBIT, MOUSE_BUTTON_FIRST, N_BUTTONS_HW, and SET_KEYBIT.

Referenced by os_inputmain().

```
403                                                                                                 {
404     if(endpoint == 2 || endpoint == -2){
405         if(urbinput[0] != 3)
406             return;
407         urbinput++;
408     }
409     for(int bit = BUTTON_HID_COUNT; bit < N_BUTTONS_HW; bit++){
410         int byte = bit / 8;
411         uchar test = 1 << (bit % 8);
412         if(urbinput[byte] & test)
413             SET_KEYBIT(kbinput, MOUSE_BUTTON_FIRST + bit);
414         else
415             CLEAR_KEYBIT(kbinput, MOUSE_BUTTON_FIRST + bit);
416     }
417 }
```

Here is the caller graph for this function:

8.26.3.3 void hid_kb_translate (unsigned char ∗ *kbinput,* int *endpoint,* int *length,* const unsigned char ∗ *urbinput*)

Definition at line 223 of file keymap.c.

References ckb_warn, CLEAR_KEYBIT, and SET_KEYBIT.

Referenced by os_inputmain().

```
223                                                                                          {
224     if(length < 1)
225         return;
226     // LUT for HID -> Corsair scancodes (-1 for no scan code, -2 for currently unsupported)
227     // Modified from Linux drivers/hid/usbhid/usbkbd.c, key codes replaced with array indices and K95 keys added
228     static const short hid_codes[256] = {
229         -1,  -1,  -1,  -1,  37,  54,  52,  39,  27,  40,  41,  42,  32,  43,  44,  45,
230         56,  55,  33,  34,  25,  28,  38,  29,  31,  53,  26,  51,  30,  50,  13,  14,
231         15,  16,  17,  18,  19,  20,  21,  22,  82,   0,  86,  24,  64,  23,  84,  35,
232         79,  80,  81,  46,  47,  12,  57,  58,  59,  36,   1,   2,   3,   4,   5,   6,
233          7,   8,   9,  10,  11,  72,  73,  74,  75,  76,  77,  78,  87,  88,  89,  95,
234         93,  94,  92, 102, 103, 104, 105, 106, 107, 115, 116, 117, 112, 113, 114, 108,
235        109, 110, 118, 119,  49,  69,  -2,  -2,  -2,  -2,  -2,  -2,  -2,  -2,  -2,  -2,
236         -2,  -2,  -2,  -2,  -2,  -2,  -2,  -2,  98,  -2,  -2,  -2,  -2,  -2,  -2,  97,
237        130, 131,  -1,  -1,  -1,  -2,  -1,  -2,  -2,  -2,  -2,  -2,  -2,  -1,  -1,  -1,
238         -2,  -2,  -2,  -2,  -2,  -1,  -1,  -1,  -1,  -1,  -1,  -1,  -1,  -1,  -1,  -1,
239         -1,  -1,  -1,  -1,  -1,  -1,  -1,  -1,  -1,  -1,  -1,  -1,  -1,  -1,  -1,  -1,
240         -1,  -1,  -1,  -1,  -1,  -1,  -1,  -1,  -1,  -1,  -1,  -1,  -1,  -1,  -1,  -1,
241         -1,  -1,  -1,  -1,  -1,  -1,  -1,  -1,  -1,  -1,  -1,  -1,  -3,  -1,  -1,  -1,  // <- -3 = non-RGB program key
242        120, 121, 122, 123, 124, 125, 126, 127, 128, 129, 136, 137, 138, 139, 140, 141,
243         60,  48,  62,  61,  91,  90,  67,  68, 142, 143,  99, 101,  -2, 130, 131,  97,
244         -2, 133, 134, 135,  -2,  96,  -2, 132,  -2,  -2,  71,  71,  71,  71,  -1,  -1,
245     };
246     switch(endpoint){
247     case 1:
248     case -1:
249         // EP 1: 6KRO input (RGB and non-RGB)
250         // Clear previous input
251         for(int i = 0; i < 256; i++){
252             if(hid_codes[i] >= 0)
253                 CLEAR_KEYBIT(kbinput, hid_codes[i]);
254         }
255         // Set new input
256         for(int i = 0; i < 8; i++){
257             if((urbinput[0] >> i) & 1)
258                 SET_KEYBIT(kbinput, hid_codes[i + 224]);
259         }
260         for(int i = 2; i < length; i++){
261             if(urbinput[i] > 3){
262                 int scan = hid_codes[urbinput[i]];
263                 if(scan >= 0)
264                     SET_KEYBIT(kbinput, scan);
265                 else
266                     ckb_warn("Got unknown key press %d on EP 1\n", urbinput[i]);
267             }
268         }
269         break;
270     case -2:
271         // EP 2 RGB: NKRO input
272         if(urbinput[0] == 1){
273             // Type 1: standard key
274             if(length != 21)
275                 return;
276             for(int bit = 0; bit < 8; bit++){
277                 if((urbinput[1] >> bit) & 1)
278                     SET_KEYBIT(kbinput, hid_codes[bit + 224]);
279                 else
280                     CLEAR_KEYBIT(kbinput, hid_codes[bit + 224]);
281             }
282             for(int byte = 0; byte < 19; byte++){
283                 char input = urbinput[byte + 2];
284                 for(int bit = 0; bit < 8; bit++){
285                     int keybit = byte * 8 + bit;
286                     int scan = hid_codes[keybit];
287                     if((input >> bit) & 1){
288                         if(scan >= 0)
289                             SET_KEYBIT(kbinput, hid_codes[keybit]);
290                         else
291                             ckb_warn("Got unknown key press %d on EP 2\n", keybit);
292                     } else if(scan >= 0)
293                         CLEAR_KEYBIT(kbinput, hid_codes[keybit]);
294                 }
295             }
296             break;
297         } else if(urbinput[0] == 2)
```

```
298             ;       // Type 2: media key (fall through)
299         else
300             break;  // No other known types
301     case 2:
302         // EP 2 Non-RGB: media keys
303         CLEAR_KEYBIT(kbinput, 97);          // mute
304         CLEAR_KEYBIT(kbinput, 98);          // stop
305         CLEAR_KEYBIT(kbinput, 99);          // prev
306         CLEAR_KEYBIT(kbinput, 100);         // play
307         CLEAR_KEYBIT(kbinput, 101);         // next
308         CLEAR_KEYBIT(kbinput, 130);         // volup
309         CLEAR_KEYBIT(kbinput, 131);         // voldn
310         for(int i = 0; i < length; i++){
311             switch(urbinput[i]){
312             case 181:
313                 SET_KEYBIT(kbinput, 101);   // next
314                 break;
315             case 182:
316                 SET_KEYBIT(kbinput, 99);    // prev
317                 break;
318             case 183:
319                 SET_KEYBIT(kbinput, 98);    // stop
320                 break;
321             case 205:
322                 SET_KEYBIT(kbinput, 100);   // play
323                 break;
324             case 226:
325                 SET_KEYBIT(kbinput, 97);    // mute
326                 break;
327             case 233:
328                 SET_KEYBIT(kbinput, 130);   // volup
329                 break;
330             case 234:
331                 SET_KEYBIT(kbinput, 131);   // voldn
332                 break;
333             }
334         }
335         break;
336     case 3:
337         // EP 3 non-RGB: NKRO input
338         if(length != 15)
339             return;
340         for(int bit = 0; bit < 8; bit++){
341             if((urbinput[0] >> bit) & 1)
342                 SET_KEYBIT(kbinput, hid_codes[bit + 224]);
343             else
344                 CLEAR_KEYBIT(kbinput, hid_codes[bit + 224]);
345         }
346         for(int byte = 0; byte < 14; byte++){
347             char input = urbinput[byte + 1];
348             for(int bit = 0; bit < 8; bit++){
349                 int keybit = byte * 8 + bit;
350                 int scan = hid_codes[keybit];
351                 if((input >> bit) & 1){
352                     if(scan >= 0)
353                         SET_KEYBIT(kbinput, hid_codes[keybit]);
354                     else
355                         ckb_warn("Got unknown key press %d on EP 3\n", keybit);
356                 } else if(scan >= 0)
357                     CLEAR_KEYBIT(kbinput, hid_codes[keybit]);
358             }
359         }
360         break;
361     }
362 }
```

Here is the caller graph for this function:

8.26.3.4 void hid_mouse_translate (unsigned char * *kbinput*, short * *xaxis*, short * *yaxis*, int *endpoint*, int *length*, const unsigned char * *urbinput*)

Definition at line 366 of file keymap.c.

References BUTTON_HID_COUNT, CLEAR_KEYBIT, MOUSE_BUTTON_FIRST, MOUSE_EXTRA_FIRST, and S-ET_KEYBIT.

Referenced by os_inputmain().

```
366
                                {
367     if((endpoint != 2 && endpoint != -2) || length < 10)
368         return;
369     // EP 2: mouse input
370     if(urbinput[0] != 1)
371         return;
372     // Byte 1 = mouse buttons (bitfield)
373     for(int bit = 0; bit < BUTTON_HID_COUNT; bit++){
374         if(urbinput[1] & (1 << bit))
375             SET_KEYBIT(kbinput, MOUSE_BUTTON_FIRST + bit);
376         else
377             CLEAR_KEYBIT(kbinput, MOUSE_BUTTON_FIRST + bit);
378     }
379     // Bytes 5 - 8: movement
380     *xaxis += *(short*)(urbinput + 5);
381     *yaxis += *(short*)(urbinput + 7);
382     // Byte 9: wheel
383     char wheel = urbinput[9];
384     if(wheel > 0)
385         SET_KEYBIT(kbinput, MOUSE_EXTRA_FIRST);          // wheelup
386     else
387         CLEAR_KEYBIT(kbinput, MOUSE_EXTRA_FIRST);
388     if(wheel < 0)
389         SET_KEYBIT(kbinput, MOUSE_EXTRA_FIRST + 1);      // wheeldn
390     else
391         CLEAR_KEYBIT(kbinput, MOUSE_EXTRA_FIRST + 1);
392 }
```

Here is the caller graph for this function:

8.26.4 Variable Documentation

8.26.4.1 const key keymap[(((152+3+12)+25)+11)]

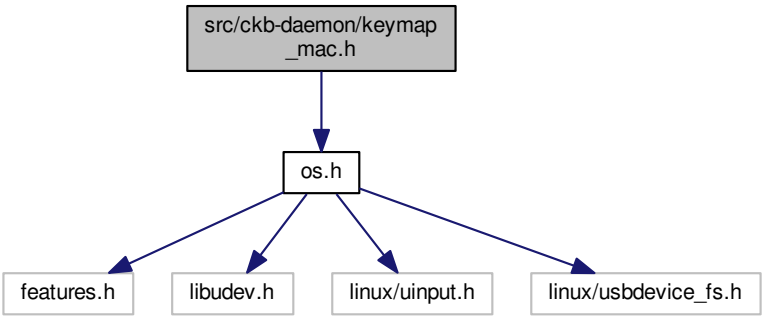
Definition at line 5 of file keymap.c.

Referenced by _cmd_get(), _cmd_macro(), cmd_bind(), cmd_rebind(), cmd_rgb(), initbind(), inputupdate_keys(), nprintkey(), printrgb(), readcmd(), and setactive_kb().

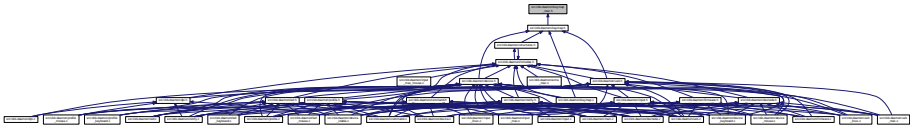
8.27 src/ckb-daemon/keymap_mac.h File Reference

```
#include "os.h"
```

Include dependency graph for keymap_mac.h:



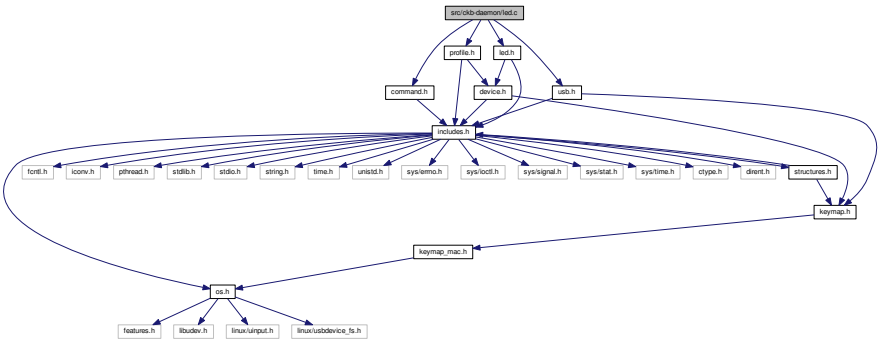
This graph shows which files directly or indirectly include this file:



8.28 src/ckb-daemon/led.c File Reference

```
#include "command.h"
#include "led.h"
#include "profile.h"
#include "usb.h"
```

Include dependency graph for led.c:



Functions

- void cmd_rgb (usbdevice ∗kb, usbmode ∗mode, int dummy, int keyindex, const char ∗code)
- static uchar iselect (const char ∗led)
- void cmd_ioff (usbdevice ∗kb, usbmode ∗mode, int dummy1, int dummy2, const char ∗led)

- void cmd_ion (usbdevice *kb, usbmode *mode, int dummy1, int dummy2, const char *led)
- void cmd_iauto (usbdevice *kb, usbmode *mode, int dummy1, int dummy2, const char *led)
- void cmd_inotify (usbdevice *kb, usbmode *mode, int nnumber, int dummy, const char *led)
- static int has_key (const char *name, const usbdevice *kb)
- char * printrgb (const lighting *light, const usbdevice *kb)

8.28.1 Function Documentation

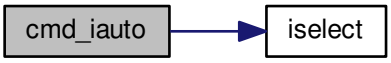
8.28.1.1 void cmd_iauto (usbdevice * *kb*, usbmode * *mode*, int *dummy1*, int *dummy2*, const char * *led*)

Definition at line 54 of file led.c.

References usbmode::ioff, usbmode::ion, iselect(), and usbdevice::vtable.

```
54                                                                                  {
55     uchar bits = iselect(led);
56     // Remove the bits from both ioff and ion
57     mode->ioff &= ~bits;
58     mode->ion &= ~bits;
59     kb->vtable->updateindicators(kb, 0);
60 }
```

Here is the call graph for this function:



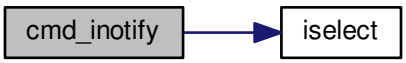
8.28.1.2 void cmd_inotify (usbdevice * *kb*, usbmode * *mode*, int *nnumber*, int *dummy*, const char * *led*)

Definition at line 62 of file led.c.

References usbmode::inotify, and iselect().

```
62                                                                                  {
63     uchar bits = iselect(led);
64     if(strstr(led, ":off"))
65         // Turn notifications for these bits off
66         mode->inotify[nnumber] &= ~bits;
67     else
68         // Turn notifications for these bits on
69         mode->inotify[nnumber] |= bits;
70 }
```

Here is the call graph for this function:



8.28.1.3 void cmd_ioff (usbdevice * kb, usbmode * mode, int dummy1, int dummy2, const char * led)

Definition at line 38 of file led.c.

References usbmode::ioff, usbmode::ion, iselect(), and usbdevice::vtable.

```
38                                                                              {
39     uchar bits = iselect(led);
40     // Add the bits to ioff, remove them from ion
41     mode->ioff |= bits;
42     mode->ion &= ~bits;
43     kb->vtable->updateindicators(kb, 0);
44 }
```

Here is the call graph for this function:

8.28.1.4 void cmd_ion (usbdevice * kb, usbmode * mode, int dummy1, int dummy2, const char * led)

Definition at line 46 of file led.c.

References usbmode::ioff, usbmode::ion, iselect(), and usbdevice::vtable.

```
46                                                                              {
47     uchar bits = iselect(led);
48     // Remove the bits from ioff, add them to ion
49     mode->ioff &= ~bits;
50     mode->ion |= bits;
51     kb->vtable->updateindicators(kb, 0);
52 }
```

Here is the call graph for this function:

8.28.1.5 void cmd_rgb (usbdevice * kb, usbmode * mode, int dummy, int keyindex, const char * code)

Definition at line 6 of file led.c.

References lighting::b, lighting::g, keymap, key::led, usbmode::light, lighting::r, and lighting::sidelight.

```
6                                                                               {
7     int index = keymap[keyindex].led;
```

```
8       if(index < 0) {
9           if (index == -2){      // Process strafe sidelights
10              uchar sideshine;
11              if (sscanf(code, "%2hhx",&sideshine)) // monochromatic
12                  mode->light.sidelight = sideshine;
13          }
14          return;
15      }
16      uchar r, g, b;
17      if(sscanf(code, "%2hhx%2hhx%2hhx", &r, &g, &b) == 3){
18          mode->light.r[index] = r;
19          mode->light.g[index] = g;
20          mode->light.b[index] = b;
21      }
22  }
```

8.28.1.6 static int has_key (const char * *name*, const usbdevice * *kb*) [static]

Definition at line 73 of file led.c.

References IS_K65, IS_K95, IS_MOUSE, IS_SABRE, IS_SCIMITAR, usbdevice::product, and usbdevice::vendor.

Referenced by printrgb().

```
73                                                      {
74      if(!name)
75          return 0;
76      if(IS_MOUSE(kb->vendor, kb->product)){
77          // Mice only have the RGB zones
78          if((IS_SABRE(kb) || IS_SCIMITAR(kb)) && !strcmp(name, "wheel"))
79              return 1;
80          if(IS_SCIMITAR(kb) && !strcmp(name, "thumb"))
81              return 1;
82          if(strstr(name, "dpi") == name || !strcmp(name, "front") || !strcmp(name, "back"))
83              return 1;
84          return 0;
85      } else {
86          // But keyboards don't have them at all
87          if(strstr(name, "dpi") == name || !strcmp(name, "front") || !strcmp(name, "back") || !strcmp(name,
        "wheel") || !strcmp(name, "thumb"))
88              return 0;
89          // Only K95 has G keys and M keys (G1 - G18, MR, M1 - M3)
90          if(!IS_K95(kb) && ((name[0] == 'g' && name[1] >= '1' && name[1] <= '9') || (name[0] == 'm' &&
         (name[1] == 'r' || name[1] == '1' || name[1] == '2' || name[1] == '3'))))
91              return 0;
92          // Only K65 has lights on VolUp/VolDn
93          if(!IS_K65(kb) && (!strcmp(name, "volup") || !strcmp(name, "voldn")))
94              return 0;
95          // K65 lacks numpad and media buttons
96          if(IS_K65(kb) && (strstr(name, "num") == name || !strcmp(name, "stop") || !strcmp(name, "prev
        ") || !strcmp(name, "play") || !strcmp(name, "next")))
97              return 0;
98      }
99      return 1;
100 }
```

Here is the caller graph for this function:

8.28.1.7 static uchar iselect (const char * *led*) [static]

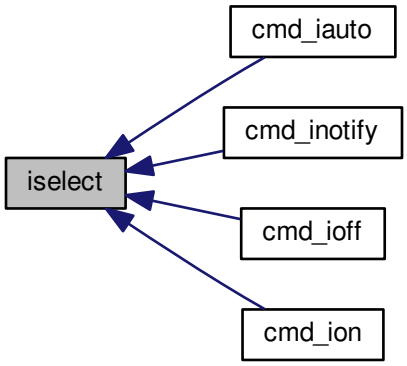
Definition at line 25 of file led.c.

References I_CAPS, I_NUM, and I_SCROLL.

Referenced by cmd_iauto(), cmd_inotify(), cmd_ioff(), and cmd_ion().

```
25                                      {
26     int result = 0;
27     if(!strncmp(led, "num", 3) || strstr(led, ",num"))
28         result |= I_NUM;
29     if(!strncmp(led, "caps", 4) || strstr(led, ",caps"))
30         result |= I_CAPS;
31     if(!strncmp(led, "scroll", 6) || strstr(led, ",scroll"))
32         result |= I_SCROLL;
33     if(!strncmp(led, "all", 3) || strstr(led, ",all"))
34         result |= I_NUM | I_CAPS | I_SCROLL;
35     return result;
36 }
```

Here is the caller graph for this function:



8.28.1.8 char* printrgb (const lighting * *light*, const usbdevice * *kb*)

Definition at line 102 of file led.c.

References lighting::b, lighting::g, has_key(), keymap, key::led, N_KEYS_EXTENDED, key::name, and lighting::r.

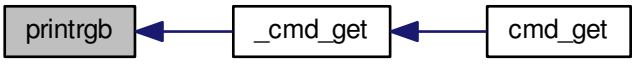
Referenced by _cmd_get().

```
102                                                             {
103     uchar r[N_KEYS_EXTENDED], g[N_KEYS_EXTENDED], b[
      N_KEYS_EXTENDED];
104     const uchar* mr = light->r;
105     const uchar* mg = light->g;
106     const uchar* mb = light->b;
107     for(int i = 0; i < N_KEYS_EXTENDED; i++){
108         // Translate the key index to an RGB index using the key map
109         int k = keymap[i].led;
110         if(k < 0)
111             continue;
112         r[i] = mr[k];
113         g[i] = mg[k];
114         b[i] = mb[k];
115     }
116     // Make a buffer to track key names and to filter out duplicates
117     char names[N_KEYS_EXTENDED][11];
118     for(int i = 0; i < N_KEYS_EXTENDED; i++){
119         const char* name = keymap[i].name;
120         if(keymap[i].led < 0 || !has_key(name, kb))
121             names[i][0] = 0;
122         else
```

```
123             strncpy(names[i], name, 11);
124     }
125     // Check to make sure these aren't all the same color
126     int same = 1;
127     for(int i = 1; i < N_KEYS_EXTENDED; i++){
128         if(!names[i][0])
129             continue;
130         if(r[i] != r[0] || g[i] != g[0] || b[i] != b[0]){
131             same = 0;
132             break;
133         }
134     }
135     // If they are, just output that color
136     if(same){
137         char* buffer = malloc(7);
138         snprintf(buffer, 7, "%02x%02x%02x", r[0], g[0], b[0]);
139         return buffer;
140     }
141     const int BUFFER_LEN = 4096;    // Should be more than enough to fit all keys
142     char* buffer = malloc(BUFFER_LEN);
143     int length = 0;
144     for(int i = 0; i < N_KEYS_EXTENDED; i++){
145         if(!names[i][0])
146             continue;
147         // Print the key name
148         int newlen = 0;
149         snprintf(buffer + length, BUFFER_LEN - length, length == 0 ? "%s%n" : " %s%n", names[i], &newlen);
150         length += newlen;
151         // Look ahead to see if any other keys have this color. If so, print them here as well.
152         uchar kr = r[i], kg = g[i], kb = b[i];
153         for(int j = i + 1; j < N_KEYS_EXTENDED; j++){
154             if(!names[j][0])
155                 continue;
156             if(r[j] != kr || g[j] != kg || b[j] != kb)
157                 continue;
158             snprintf(buffer + length, BUFFER_LEN - length, ",%s%n", names[j], &newlen);
159             length += newlen;
160             // Erase the key's name so it won't get printed later
161             names[j][0] = 0;
162         }
163         // Print the color
164         snprintf(buffer + length, BUFFER_LEN - length, ":%02x%02x%02x%n", kr, kg, kb, &newlen);
165         length += newlen;
166     }
167     return buffer;
168 }
```

Here is the call graph for this function:

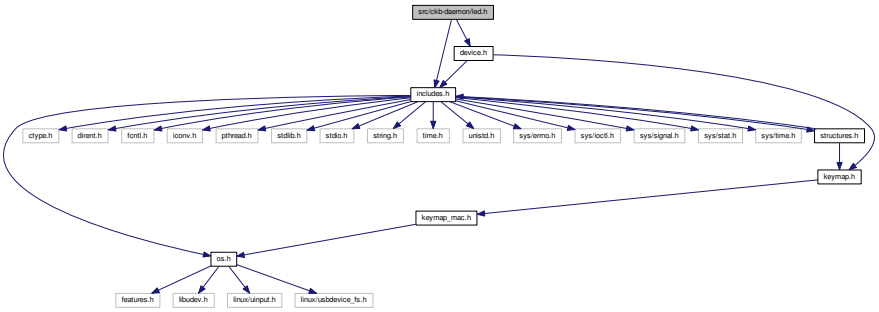
Here is the caller graph for this function:



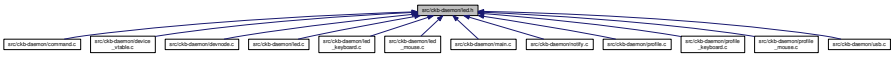
8.29 src/ckb-daemon/led.h File Reference

```
#include "includes.h"
#include "device.h"
```

Include dependency graph for led.h:



This graph shows which files directly or indirectly include this file:



Functions

- int updatergb_kb (usbdevice ∗kb, int force)
- int updatergb_mouse (usbdevice ∗kb, int force)
- int savergb_kb (usbdevice ∗kb, lighting ∗light, int mode)
- int savergb_mouse (usbdevice ∗kb, lighting ∗light, int mode)
- int loadrgb_kb (usbdevice ∗kb, lighting ∗light, int mode)
- int loadrgb_mouse (usbdevice ∗kb, lighting ∗light, int mode)
- char ∗ printrgb (const lighting ∗light, const usbdevice ∗kb)
- void cmd_rgb (usbdevice ∗kb, usbmode ∗mode, int dummy, int keyindex, const char ∗code)
- void cmd_ioff (usbdevice ∗kb, usbmode ∗mode, int dummy1, int dummy2, const char ∗led)
- void cmd_ion (usbdevice ∗kb, usbmode ∗mode, int dummy1, int dummy2, const char ∗led)
- void cmd_iauto (usbdevice ∗kb, usbmode ∗mode, int dummy1, int dummy2, const char ∗led)
- void cmd_inotify (usbdevice ∗kb, usbmode ∗mode, int nnumber, int dummy, const char ∗led)

8.29.1 Function Documentation

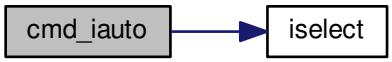
8.29.1.1 void cmd_iauto (usbdevice ∗ *kb*, usbmode ∗ *mode*, int *dummy1*, int *dummy2*, const char ∗ *led*)

Definition at line 54 of file led.c.

References usbmode::ioff, usbmode::ion, iselect(), and usbdevice::vtable.

```
54                                                                                  {
55     uchar bits = iselect(led);
56     // Remove the bits from both ioff and ion
57     mode->ioff &= ~bits;
58     mode->ion &= ~bits;
59     kb->vtable->updateindicators(kb, 0);
60 }
```

Here is the call graph for this function:



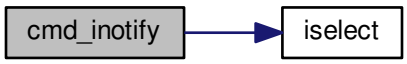
8.29.1.2 void cmd_inotify (usbdevice * *kb*, usbmode * *mode*, int *nnumber*, int *dummy*, const char * *led*)

Definition at line 62 of file led.c.

References usbmode::inotify, and iselect().

```
62                                                                                  {
63     uchar bits = iselect(led);
64     if(strstr(led, ":off"))
65         // Turn notifications for these bits off
66         mode->inotify[nnumber] &= ~bits;
67     else
68         // Turn notifications for these bits on
69         mode->inotify[nnumber] |= bits;
70 }
```

Here is the call graph for this function:



8.29.1.3 void cmd_ioff (usbdevice * *kb*, usbmode * *mode*, int *dummy1*, int *dummy2*, const char * *led*)

Definition at line 38 of file led.c.

References usbmode::ioff, usbmode::ion, iselect(), and usbdevice::vtable.

```
38                                                                                  {
39     uchar bits = iselect(led);
40     // Add the bits to ioff, remove them from ion
41     mode->ioff |= bits;
42     mode->ion &= ~bits;
43     kb->vtable->updateindicators(kb, 0);
44 }
```

Here is the call graph for this function:

8.29.1.4 void cmd_ion (usbdevice * kb, usbmode * mode, int dummy1, int dummy2, const char * led)

Definition at line 46 of file led.c.

References usbmode::ioff, usbmode::ion, iselect(), and usbdevice::vtable.

```
46                                                                                  {
47     uchar bits = iselect(led);
48     // Remove the bits from ioff, add them to ion
49     mode->ioff &= ~bits;
50     mode->ion |= bits;
51     kb->vtable->updateindicators(kb, 0);
52 }
```

Here is the call graph for this function:

8.29.1.5 void cmd_rgb (usbdevice * kb, usbmode * mode, int dummy, int keyindex, const char * code)

Definition at line 6 of file led.c.

References lighting::b, lighting::g, keymap, key::led, usbmode::light, lighting::r, and lighting::sidelight.

```
6                                                                                   {
7     int index = keymap[keyindex].led;
8     if(index < 0) {
9         if (index == -2){     // Process strafe sidelights
10            uchar sideshine;
11            if (sscanf(code, "%2hhx",&sideshine)) // monochromatic
12                mode->light.sidelight = sideshine;
13        }
14        return;
15    }
16    uchar r, g, b;
17    if(sscanf(code, "%2hhx%2hhx%2hhx", &r, &g, &b) == 3){
18        mode->light.r[index] = r;
19        mode->light.g[index] = g;
20        mode->light.b[index] = b;
21    }
22 }
```

8.29.1.6 int loadrgb_kb (usbdevice * *kb,* lighting * *light,* int *mode*)

Since Firmware Version 2.05 the answers for getting the stored color-maps from the hardware has changed a bit. So comparing for the correct answer cannot validate against the cmd, and has to be done against a third map.

Definition at line 181 of file led_keyboard.c.

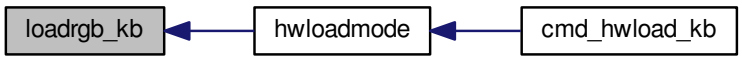
References lighting::b, ckb_err, usbdevice::fwversion, lighting::g, MSG_SIZE, N_KEYS_HW, lighting::r, usbrecv, and usbsend.

Referenced by hwloadmode().

```
181                                                     {
182     if(kb->fwversion >= 0x0120){
183         uchar data_pkt[12][MSG_SIZE] = {
184             { 0x0e, 0x14, 0x03, 0x01, 0x01, mode + 1, 0x01 },
185             { 0xff, 0x01, 60, 0 },
186             { 0xff, 0x02, 60, 0 },
187             { 0xff, 0x03, 24, 0 },
188             { 0x0e, 0x14, 0x03, 0x01, 0x01, mode + 1, 0x02 },
189             { 0xff, 0x01, 60, 0 },
190             { 0xff, 0x02, 60, 0 },
191             { 0xff, 0x03, 24, 0 },
192             { 0x0e, 0x14, 0x03, 0x01, 0x01, mode + 1, 0x03 },
193             { 0xff, 0x01, 60, 0 },
194             { 0xff, 0x02, 60, 0 },
195             { 0xff, 0x03, 24, 0 },
196         };
197         uchar in_pkt[4][MSG_SIZE] = {
198             { 0x0e, 0x14, 0x03, 0x01 },
199             { 0xff, 0x01, 60, 0 },
200             { 0xff, 0x02, 60, 0 },
201             { 0xff, 0x03, 24, 0 },
202         };
203
207
208         uchar cmp_pkt[4][4] = {
209             { 0x0e, 0x14, 0x03, 0x01 },
210             { 0x0e, 0xff, 0x01, 60 },
211             { 0x0e, 0xff, 0x02, 60 },
212             { 0x0e, 0xff, 0x03, 24 },
213         };
214         // Read colors
215         uchar* colors[3] = { light->r, light->g, light->b };
216         for(int clr = 0; clr < 3; clr++){
217             for(int i = 0; i < 4; i++){
218                 if(!usbrecv(kb, data_pkt[i + clr * 4], in_pkt[i]))
219                     return -1;
220                 // Make sure the first four bytes match
221                 // see comment above
222                 // if(memcmp(p, data_pkt[i + clr * 4], 4)){
223                 if (memcmp(in_pkt[i], (kb->fwversion >= 0x0205)? cmp_pkt[i] : data_pkt[i + clr * 4
    ], 4)) {
224                     ckb_err("Bad input header\n");
225                     ckb_err("color = %d, i = %d, mode = %d\nInput(Antwort): %2.2x %2.2x %2.2x %2.2x
     %2.2x %2.2x %2.2x %2.2x\nOutput (Frage): %2.2x %2.2x %2.2x %2.2x\n", clr, i, mode,
226                         in_pkt[i][0], in_pkt[i][1], in_pkt[i][2], in_pkt[i][3], in_pkt[i][4], in_pkt[i][5],
     in_pkt[i][6], in_pkt[i][7],
227                         // data_pkt[i + clr * 4][0],    data_pkt[i + clr * 4 ][1],  data_pkt[i + clr *
     4 ][2],  data_pkt[i + clr * 4 ][3]);
228                         cmp_pkt[i][0], cmp_pkt[i][1], cmp_pkt[i][2], cmp_pkt[i][3]);
229                     in_pkt[2][0] = 0x99;
230                     in_pkt[2][1] = 0x99;
231                     in_pkt[2][2] = 0x99;
232                     in_pkt[2][3] = 0x99;
233                     usbrecv(kb, in_pkt[2], in_pkt[2]); // just to find it in the wireshark log
234                     return -1;
235                 }
236             }
237             // Copy colors to lighting. in_pkt[0] is irrelevant.
238             memcpy(colors[clr], in_pkt[1] + 4, 60);
239             memcpy(colors[clr] + 60, in_pkt[2] + 4, 60);
240             memcpy(colors[clr] + 120, in_pkt[3] + 4, 24);
241         }
242     } else {
243         uchar data_pkt[5][MSG_SIZE] = {
244             { 0x0e, 0x14, 0x02, 0x01, 0x01, mode + 1, 0 },
245             { 0xff, 0x01, 60, 0 },
246             { 0xff, 0x02, 60, 0 },
247             { 0xff, 0x03, 60, 0 },
248             { 0xff, 0x04, 36, 0 },
249         };
250         uchar in_pkt[4][MSG_SIZE] = {
251             { 0xff, 0x01, 60, 0 },
```

```
252             { 0xff, 0x02, 60, 0 },
253             { 0xff, 0x03, 60, 0 },
254             { 0xff, 0x04, 36, 0 },
255         };
256         // Write initial packet
257         if(!usbsend(kb, data_pkt[0], 1))
258             return -1;
259         // Read colors
260         for(int i = 1; i < 5; i++){
261             if(!usbrecv(kb, data_pkt[i],in_pkt[i - 1]))
262                 return -1;
263             if(memcmp(in_pkt[i - 1], data_pkt[i], 4)){
264                 ckb_err("Bad input header\n");
265                 return -1;
266             }
267         }
268         // Copy the data back to the mode
269         uint8_t mr[N_KEYS_HW / 2], mg[N_KEYS_HW / 2], mb[
    N_KEYS_HW / 2];
270         memcpy(mr,      in_pkt[0] +  4, 60);
271         memcpy(mr + 60, in_pkt[1] +  4, 12);
272         memcpy(mg,      in_pkt[1] + 16, 48);
273         memcpy(mg + 48, in_pkt[2] +  4, 24);
274         memcpy(mb,      in_pkt[2] + 28, 36);
275         memcpy(mb + 36, in_pkt[3] +  4, 36);
276         // Unpack LED data to 8bpc format
277         for(int i = 0; i < N_KEYS_HW; i++){
278             int     i_2 = i / 2;
279             uint8_t r, g, b;
280
281             // 3-bit intensities stored in alternate nybbles.
282             if (i & 1) {
283                 r = 7 - (mr[i_2] >> 4);
284                 g = 7 - (mg[i_2] >> 4);
285                 b = 7 - (mb[i_2] >> 4);
286             } else {
287                 r = 7 - (mr[i_2] & 0x0F);
288                 g = 7 - (mg[i_2] & 0x0F);
289                 b = 7 - (mb[i_2] & 0x0F);
290             }
291             // Scale 3-bit values up to 8 bits.
292             light->r[i] = r << 5 | r << 2 | r >> 1;
293             light->g[i] = g << 5 | g << 2 | g >> 1;
294             light->b[i] = b << 5 | b << 2 | b >> 1;
295         }
296     }
297     return 0;
298 }
```

Here is the caller graph for this function:



8.29.1.7 int loadrgb_mouse (usbdevice * *kb*, lighting * *light*, int *mode*)

Definition at line 81 of file led_mouse.c.

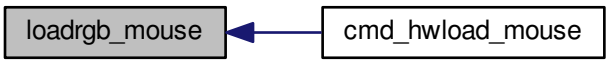
References lighting::b, ckb_err, lighting::g, IS_SABRE, IS_SCIMITAR, LED_DPI, LED_MOUSE, MSG_SIZE, lighting::r, and usbrecv.

Referenced by cmd_hwload_mouse().

```
81                                                      {
82     uchar data_pkt[MSG_SIZE] = { 0x0e, 0x13, 0x10, 1, 0 };
83     uchar in_pkt[MSG_SIZE] = { 0 };
84     // Load each RGB zone
85     int zonecount = IS_SCIMITAR(kb) ? 4 : IS_SABRE(kb) ? 3 : 2;
```

```
86      for(int i = 0; i < zonecount; i++){
87          if(!usbrecv(kb, data_pkt, in_pkt))
88              return -1;
89          if(memcmp(in_pkt, data_pkt, 4)){
90              ckb_err("Bad input header\n");
91              return -2;
92          }
93          // Copy data
94          int led = LED_MOUSE + i;
95          if(led >= LED_DPI)
96              led++;          // Skip DPI light
97          light->r[led] = in_pkt[4];
98          light->g[led] = in_pkt[5];
99          light->b[led] = in_pkt[6];
100          // Set packet for next zone
101          data_pkt[2]++;
102      }
103      return 0;
104  }
```

Here is the caller graph for this function:



8.29.1.8 char* printrgb (const lighting * *light*, const usbdevice * *kb*)

Definition at line 102 of file led.c.

References lighting::b, lighting::g, has_key(), keymap, key::led, N_KEYS_EXTENDED, key::name, and lighting::r.

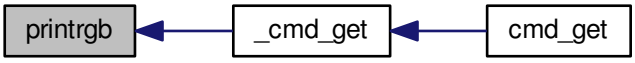
Referenced by _cmd_get().

```
102                                                     {
103      uchar r[N_KEYS_EXTENDED], g[N_KEYS_EXTENDED], b[
       N_KEYS_EXTENDED];
104      const uchar* mr = light->r;
105      const uchar* mg = light->g;
106      const uchar* mb = light->b;
107      for(int i = 0; i < N_KEYS_EXTENDED; i++){
108          // Translate the key index to an RGB index using the key map
109          int k = keymap[i].led;
110          if(k < 0)
111              continue;
112          r[i] = mr[k];
113          g[i] = mg[k];
114          b[i] = mb[k];
115      }
116      // Make a buffer to track key names and to filter out duplicates
117      char names[N_KEYS_EXTENDED][11];
118      for(int i = 0; i < N_KEYS_EXTENDED; i++){
119          const char* name = keymap[i].name;
120          if(keymap[i].led < 0 || !has_key(name, kb))
121              names[i][0] = 0;
122          else
123              strncpy(names[i], name, 11);
124      }
125      // Check to make sure these aren't all the same color
126      int same = 1;
127      for(int i = 1; i < N_KEYS_EXTENDED; i++){
128          if(!names[i][0])
129              continue;
130          if(r[i] != r[0] || g[i] != g[0] || b[i] != b[0]){
131              same = 0;
132              break;
133          }
134      }
```

```
135     // If they are, just output that color
136     if(same){
137         char* buffer = malloc(7);
138         snprintf(buffer, 7, "%02x%02x%02x", r[0], g[0], b[0]);
139         return buffer;
140     }
141     const int BUFFER_LEN = 4096;    // Should be more than enough to fit all keys
142     char* buffer = malloc(BUFFER_LEN);
143     int length = 0;
144     for(int i = 0; i < N_KEYS_EXTENDED; i++){
145         if(!names[i][0])
146             continue;
147         // Print the key name
148         int newlen = 0;
149         snprintf(buffer + length, BUFFER_LEN - length, length == 0 ? "%s%n" : " %s%n", names[i], &newlen);
150         length += newlen;
151         // Look ahead to see if any other keys have this color. If so, print them here as well.
152         uchar kr = r[i], kg = g[i], kb = b[i];
153         for(int j = i + 1; j < N_KEYS_EXTENDED; j++){
154             if(!names[j][0])
155                 continue;
156             if(r[j] != kr || g[j] != kg || b[j] != kb)
157                 continue;
158             snprintf(buffer + length, BUFFER_LEN - length, ",%s%n", names[j], &newlen);
159             length += newlen;
160             // Erase the key's name so it won't get printed later
161             names[j][0] = 0;
162         }
163         // Print the color
164         snprintf(buffer + length, BUFFER_LEN - length, ":%02x%02x%02x%n", kr, kg, kb, &newlen);
165         length += newlen;
166     }
167     return buffer;
168 }
```

Here is the call graph for this function:

Here is the caller graph for this function:



8.29.1.9 int savergb_kb (usbdevice * *kb*, lighting * *light*, int *mode*)

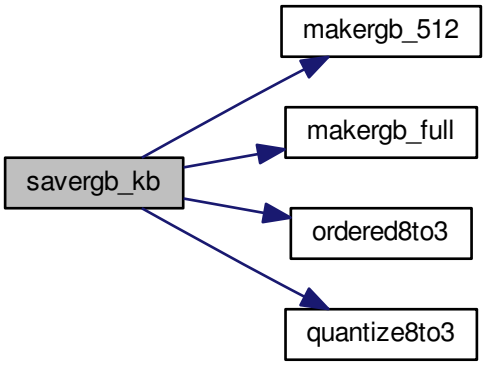
Definition at line 139 of file led_keyboard.c.

References usbdevice::dither, usbdevice::fwversion, IS_STRAFE, makergb_512(), makergb_full(), MSG_SIZE, ordered8to3(), quantize8to3(), and usbsend.

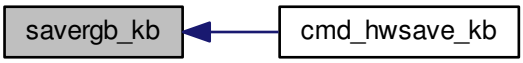
Referenced by cmd_hwsave_kb().

```
139                                                     {
140     if(kb->fwversion >= 0x0120){
141         uchar data_pkt[12][MSG_SIZE] = {
142             // Red
143             { 0x7f, 0x01, 60, 0 },
144             { 0x7f, 0x02, 60, 0 },
145             { 0x7f, 0x03, 24, 0 },
146             { 0x07, 0x14, 0x03, 0x01, 0x01, mode + 1, 0x01 },
147             // Green
148             { 0x7f, 0x01, 60, 0 },
149             { 0x7f, 0x02, 60, 0 },
150             { 0x7f, 0x03, 24, 0 },
151             { 0x07, 0x14, 0x03, 0x01, 0x01, mode + 1, 0x02 },
152             // Blue
153             { 0x7f, 0x01, 60, 0 },
154             { 0x7f, 0x02, 60, 0 },
155             { 0x7f, 0x03, 24, 0 },
156             { 0x07, 0x14, 0x03, 0x01, 0x01, mode + 1, 0x03 }
157         };
158         makergb_full(light, data_pkt);
159         if(!usbsend(kb, data_pkt[0], 12))
160             return -1;
161         if (IS_STRAFE(kb)){ // end save
162             uchar save_end_pkt[MSG_SIZE] = { 0x07, 0x14, 0x04, 0x01, 0x01 };
163             if(!usbsend(kb, save_end_pkt, 1))
164                 return -1;
165         }
166     } else {
167         uchar data_pkt[5][MSG_SIZE] = {
168             { 0x7f, 0x01, 60, 0 },
169             { 0x7f, 0x02, 60, 0 },
170             { 0x7f, 0x03, 60, 0 },
171             { 0x7f, 0x04, 36, 0 },
172             { 0x07, 0x14, 0x02, 0x00, 0x01, mode + 1 }
173         };
174         makergb_512(light, data_pkt, kb->dither ? ordered8to3 :
    quantize8to3);
175         if(!usbsend(kb, data_pkt[0], 5))
176             return -1;
177     }
178     return 0;
179 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



8.29.1.10 int savergb_mouse (usbdevice * *kb*, lighting * *light*, int *mode*)

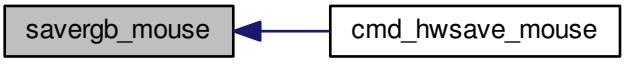
Definition at line 62 of file led_mouse.c.

References lighting::b, lighting::g, IS_SABRE, IS_SCIMITAR, LED_DPI, LED_MOUSE, MSG_SIZE, lighting::r, and usbsend.

Referenced by cmd_hwsave_mouse().

```
62                                                      {
63     uchar data_pkt[MSG_SIZE] = { 0x07, 0x13, 0x10, 1, 0 };
64     // Save each RGB zone, minus the DPI light which is sent in the DPI packets
65     int zonecount = IS_SCIMITAR(kb) ? 4 : IS_SABRE(kb) ? 3 : 2;
66     for(int i = 0; i < zonecount; i++){
67         int led = LED_MOUSE + i;
68         if(led >= LED_DPI)
69             led++;          // Skip DPI light
70         data_pkt[4] = light->r[led];
71         data_pkt[5] = light->g[led];
72         data_pkt[6] = light->b[led];
73         if(!usbsend(kb, data_pkt, 1))
74             return -1;
75         // Set packet for next zone
76         data_pkt[2]++;
77     }
78     return 0;
79 }
```

Here is the caller graph for this function:



8.29.1.11 int updatergb_kb (usbdevice * *kb*, int *force*)

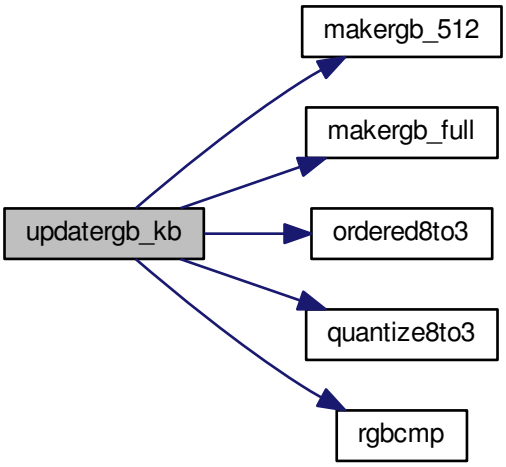
Definition at line 77 of file led_keyboard.c.

References usbdevice::active, usbprofile::currentmode, usbdevice::dither, lighting::forceupdate, IS_FULLRANGE, usbprofile::lastlight, usbmode::light, makergb_512(), makergb_full(), MSG_SIZE, ordered8to3(), usbdevice::profile, quantize8to3(), rgbcmp(), lighting::sidelight, and usbsend.

```
77                                          {
```

```
78      if(!kb->active)
79          return 0;
80      lighting* lastlight = &kb->profile->lastlight;
81      lighting* newlight = &kb->profile->currentmode->
    light;
82      // Don't do anything if the lighting hasn't changed
83      if(!force && !lastlight->forceupdate && !newlight->forceupdate
84              && !rgbcmp(lastlight, newlight) && lastlight->sidelight == newlight->
    sidelight)   // strafe sidelights
85          return 0;
86      lastlight->forceupdate = newlight->forceupdate = 0;
87
88      if(IS_FULLRANGE(kb)){
89          // Update strafe sidelights if necessary
90          if(lastlight->sidelight != newlight->sidelight) {
91              uchar data_pkt[2][MSG_SIZE] = {
92                   { 0x07, 0x05, 0x08, 0x00, 0x00 },
93                   { 0x07, 0x05, 0x02, 0, 0x03 }
94               };
95               if (newlight->sidelight)
96                   data_pkt[0][4]=1;    // turn on
97               if(!usbsend(kb, data_pkt[0], 2))
98                   return -1;
99          }
100          // 16.8M color lighting works fine on strafe and is the only way it actually works
101          uchar data_pkt[12][MSG_SIZE] = {
102              // Red
103              { 0x7f, 0x01, 0x3c, 0 },
104              { 0x7f, 0x02, 0x3c, 0 },
105              { 0x7f, 0x03, 0x18, 0 },
106              { 0x07, 0x28, 0x01, 0x03, 0x01, 0},
107              // Green
108              { 0x7f, 0x01, 0x3c, 0 },
109              { 0x7f, 0x02, 0x3c, 0 },
110              { 0x7f, 0x03, 0x18, 0 },
111              { 0x07, 0x28, 0x02, 0x03, 0x01, 0},
112              // Blue
113              { 0x7f, 0x01, 0x3c, 0 },
114              { 0x7f, 0x02, 0x3c, 0 },
115              { 0x7f, 0x03, 0x18, 0 },
116              { 0x07, 0x28, 0x03, 0x03, 0x02, 0}
117          };
118          makergb_full(newlight, data_pkt);
119          if(!usbsend(kb, data_pkt[0], 12))
120              return -1;
121      } else {
122          // On older keyboards it looks flickery and causes lighting glitches, so we don't use it.
123          uchar data_pkt[5][MSG_SIZE] = {
124              { 0x7f, 0x01, 60, 0 },
125              { 0x7f, 0x02, 60, 0 },
126              { 0x7f, 0x03, 60, 0 },
127              { 0x7f, 0x04, 36, 0 },
128              { 0x07, 0x27, 0x00, 0x00, 0xD8 }
129          };
130          makergb_512(newlight, data_pkt, kb->dither ?
    ordered8to3 : quantize8to3);
131          if(!usbsend(kb, data_pkt[0], 5))
132              return -1;
133      }
134
135      memcpy(lastlight, newlight, sizeof(lighting));
136      return 0;
137  }
```

Here is the call graph for this function:



8.29.1.12 int updatergb_mouse (usbdevice * *kb*, int *force*)

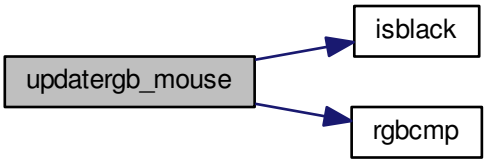
Definition at line 20 of file led_mouse.c.

References usbdevice::active, lighting::b, usbprofile::currentmode, lighting::forceupdate, lighting::g, isblack(), usbprofile::lastlight, LED_MOUSE, usbmode::light, MSG_SIZE, N_MOUSE_ZONES, usbdevice::profile, lighting::r, rgbcmp(), and usbsend.

```
20                                              {
21     if(!kb->active)
22         return 0;
23     lighting* lastlight = &kb->profile->lastlight;
24     lighting* newlight = &kb->profile->currentmode->
       light;
25     // Don't do anything if the lighting hasn't changed
26     if(!force && !lastlight->forceupdate && !newlight->forceupdate
27             && !rgbcmp(lastlight, newlight))
28         return 0;
29     lastlight->forceupdate = newlight->forceupdate = 0;
30
31     // Send the RGB values for each zone to the mouse
32     uchar data_pkt[2][MSG_SIZE] = {
33         { 0x07, 0x22, N_MOUSE_ZONES, 0x01, 0 }, // RGB colors
34         { 0x07, 0x05, 0x02, 0 }                 // Lighting on/off
35     };
36     uchar* rgb_data = &data_pkt[0][4];
37     for(int i = 0; i < N_MOUSE_ZONES; i++){
38         *rgb_data++ = i + 1;
39         *rgb_data++ = newlight->r[LED_MOUSE + i];
40         *rgb_data++ = newlight->g[LED_MOUSE + i];
41         *rgb_data++ = newlight->b[LED_MOUSE + i];
42     }
43     // Send RGB data
44     if(!usbsend(kb, data_pkt[0], 1))
45         return -1;
46     int was_black = isblack(kb, lastlight), is_black = isblack(kb, newlight);
47     if(is_black){
48         // If the lighting is black, send the deactivation packet (M65 only)
49         if(!usbsend(kb, data_pkt[1], 1))
50             return -1;
51     } else if(was_black || force){
52         // If the lighting WAS black, or if we're on forced update, send the activation packet
53         data_pkt[1][4] = 1;
54         if(!usbsend(kb, data_pkt[1], 1))
```

```
55             return -1;
56     }
57
58     memcpy(lastlight, newlight, sizeof(lighting));
59     return 0;
60 }
```

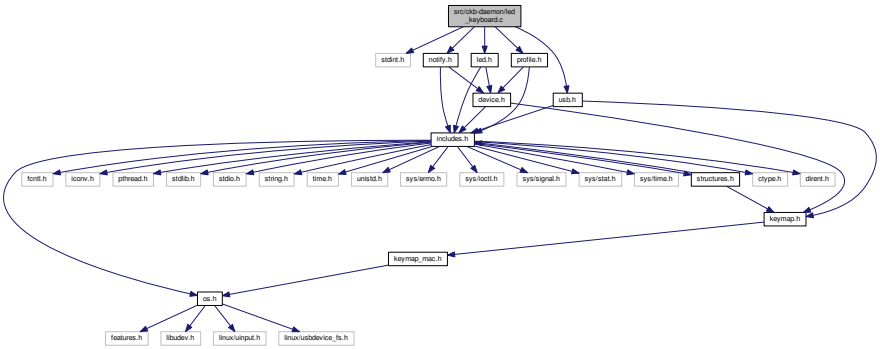
Here is the call graph for this function:



8.30 src/ckb-daemon/led_keyboard.c File Reference

```
#include <stdint.h>
#include "led.h"
#include "notify.h"
#include "profile.h"
#include "usb.h"
```

Include dependency graph for led_keyboard.c:



Macros

- #define BR1(x) ((((x) & 0xaa) >> 1) | (((x) & 0x55) << 1))
- #define BR2(x) (((BR1(x) & 0xcc) >> 2) | ((BR1(x) & 0x33) << 2))
- #define BR4(x) (((BR2(x) & 0xf0) >> 4) | ((BR2(x) & 0x0f) << 4))
- #define O0(i) BR4(i),
- #define O1(i) O0(i) O0((i) + 1)
- #define O2(i) O1(i) O1((i) + 2)
- #define O3(i) O2(i) O2((i) + 4)
- #define O4(i) O3(i) O3((i) + 8)
- #define O5(i) O4(i) O4((i) + 16)

- #define O6(i) O5(i) O5((i) + 32)
- #define O7(i) O6(i) O6((i) + 64)
- #define O8(i) O7(i) O7((i) + 127)

Functions

- static uchar ordered8to3 (int index, uchar value)
- static uchar quantize8to3 (int index, uchar value)
- static void makergb_512 (const lighting *light, uchar data_pkt[5][64], uchar(*ditherfn)(int, uchar))
- static void makergb_full (const lighting *light, uchar data_pkt[12][64])
- static int rgbcmp (const lighting *lhs, const lighting *rhs)
- int updatergb_kb (usbdevice *kb, int force)
- int savergb_kb (usbdevice *kb, lighting *light, int mode)
- int loadrgb_kb (usbdevice *kb, lighting *light, int mode)

Variables

- static uchar bit_reverse_table [256] = { ((((((((((0) & 0xaa) >> 1) | (((0) & 0x55) << 1)) & 0xcc) >> 2) | ((((((0) & 0xaa) >> 1) | (((0) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((0) & 0xaa) >> 1) | (((0) & 0x55) << 1)) & 0xcc) >> 2) | ((((((0) & 0xaa) >> 1) | (((0) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((0) + 1) & 0xaa) >> 1) | ((((0) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((0) + 1) & 0xaa) >> 1) | ((((0) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((0) + 1) & 0xaa) >> 1) | ((((0) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((0) + 1) & 0xaa) >> 1) | ((((0) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((0) + 2) & 0xaa) >> 1) | ((((0) + 2) & 0x55) << 1)) & 0xcc) >> 2) | ((((((0) + 2) & 0xaa) >> 1) | ((((0) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((0) + 2) & 0xaa) >> 1) | ((((0) + 2) & 0x55) << 1)) & 0xcc) >> 2) | ((((((0) + 2) & 0xaa) >> 1) | ((((0) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((0) + 2) + 1) & 0xaa) >> 1) | (((((0) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((0) + 2) + 1) & 0xaa) >> 1) | (((((0) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((0) + 2) + 1) & 0xaa) >> 1) | (((((0) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((0) + 2) + 1) & 0xaa) >> 1) | (((((0) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((0) + 4) & 0xaa) >> 1) | ((((0) + 4) & 0x55) << 1)) & 0xcc) >> 2) | ((((((0) + 4) & 0xaa) >> 1) | ((((0) + 4) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((0) + 4) & 0xaa) >> 1) | ((((0) + 4) & 0x55) << 1)) & 0xcc) >> 2) | ((((((0) + 4) & 0xaa) >> 1) | ((((0) + 4) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((0) + 4) + 1) & 0xaa) >> 1) | (((((0) + 4) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((0) + 4) + 1) & 0xaa) >> 1) | (((((0) + 4) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((0) + 4) + 1) & 0xaa) >> 1) | (((((0) + 4) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((0) + 4) + 1) & 0xaa) >> 1) | (((((0) + 4) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((0) + 4) + 2) & 0xaa) >> 1) | (((((0) + 4) + 2) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((0) + 4) + 2) & 0xaa) >> 1) | (((((0) + 4) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((0) + 4) + 2) & 0xaa) >> 1) | (((((0) + 4) + 2) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((0) + 4) + 2) & 0xaa) >> 1) | (((((0) + 4) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((((0) + 4) + 2) + 1) & 0xaa) >> 1) | ((((((0) + 4) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((0) + 4) + 2) + 1) & 0xaa) >> 1) | ((((((0) + 4) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((0) + 4) + 2) + 1) & 0xaa) >> 1) | ((((((0) + 4) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((0) + 4) + 2) + 1) & 0xaa) >> 1) | ((((((0) + 4) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((0) + 8) & 0xaa) >> 1) | ((((0) + 8) & 0x55) << 1)) & 0xcc) >> 2) | ((((((0) + 8) & 0xaa) >> 1) | ((((0) + 8) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((0) + 8) & 0xaa) >> 1) | ((((0) + 8) & 0x55) << 1)) & 0xcc) >> 2) | ((((((0) + 8) & 0xaa) >> 1) | ((((0) + 8) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((0) + 8) + 1) & 0xaa) >> 1) | (((((0) + 8) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((0) + 8) + 1) & 0xaa) >> 1) | (((((0) + 8) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((0) + 8) + 1) & 0xaa) >> 1) | (((((0) + 8) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((0) + 8) + 1) & 0xaa) >> 1) | (((((0) + 8) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((0) + 8) + 2) & 0xaa) >> 1) | (((((0) + 8) + 2) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((0) + 8) + 2) & 0xaa) >> 1) | (((((0) + 8) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((0) + 8) + 2) & 0xaa) >> 1) | (((((0) + 8) + 2) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((0) + 8) + 2) & 0xaa) >> 1) | (((((0) + 8) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((((0) + 8) + 2) + 1) & 0xaa) >>

1) | (((((((0) + 8) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((0) + 8) + 2) + 1) & 0xaa) >> 1) | ((((((0) + 8) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((0) + 8) + 2) + 1) & 0xaa) >> 1) | ((((((0) + 8) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((0) + 8) + 2) + 1) & 0xaa) >> 1) | ((((((0) + 8) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((((0) + 8) + 4) & 0xaa) >> 1) | (((((0) + 8) + 4) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((0) + 8) + 4) & 0xaa) >> 1) | (((((0) + 8) + 4) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((((0) + 8) + 4) & 0xaa) >> 1) | (((((0) + 8) + 4) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((0) + 8) + 4) & 0xaa) >> 1) | (((((0) + 8) + 4) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((((0) + 8) + 4) + 1) & 0xaa) >> 1) | ((((((0) + 8) + 4) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((0) + 8) + 4) + 1) & 0xaa) >> 1) | ((((((0) + 8) + 4) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((0) + 8) + 4) + 1) & 0xaa) >> 1) | ((((((0) + 8) + 4) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((0) + 8) + 4) + 1) & 0xaa) >> 1) | ((((((0) + 8) + 4) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((((0) + 8) + 4) + 2) & 0xaa) >> 1) | ((((((0) + 8) + 4) + 2) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((0) + 8) + 4) + 2) & 0xaa) >> 1) | ((((((0) + 8) + 4) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((0) + 8) + 4) + 2) & 0xaa) >> 1) | ((((((0) + 8) + 4) + 2) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((0) + 8) + 4) + 2) & 0xaa) >> 1) | ((((((0) + 8) + 4) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((((((0) + 8) + 4) + 2) + 1) & 0xaa) >> 1) | (((((((0) + 8) + 4) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((0) + 8) + 4) + 2) + 1) & 0xaa) >> 1) | (((((((0) + 8) + 4) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((((((0) + 8) + 4) + 2) + 1) & 0xaa) >> 1) | (((((((0) + 8) + 4) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((0) + 8) + 4) + 2) + 1) & 0xaa) >> 1) | (((((((0) + 8) + 4) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((0) + 16) & 0xaa) >> 1) | ((((0) + 16) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((0) + 16) & 0xaa) >> 1) | ((((0) + 16) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((0) + 16) & 0xaa) >> 1) | ((((0) + 16) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((0) + 16) & 0xaa) >> 1) | ((((0) + 16) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((0) + 16) + 1) & 0xaa) >> 1) | ((((0) + 16) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((0) + 16) + 1) & 0xaa) >> 1) | ((((0) + 16) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((0) + 16) + 1) & 0xaa) >> 1) | ((((0) + 16) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((0) + 16) + 1) & 0xaa) >> 1) | ((((0) + 16) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((0) + 16) + 2) & 0xaa) >> 1) | ((((0) + 16) + 2) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((0) + 16) + 2) & 0xaa) >> 1) | ((((0) + 16) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((0) + 16) + 2) & 0xaa) >> 1) | ((((0) + 16) + 2) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((0) + 16) + 2) & 0xaa) >> 1) | ((((0) + 16) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((((0) + 16) + 2) + 1) & 0xaa) >> 1) | (((((0) + 16) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((0) + 16) + 2) + 1) & 0xaa) >> 1) | (((((0) + 16) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((((0) + 16) + 2) + 1) & 0xaa) >> 1) | (((((0) + 16) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((0) + 16) + 2) + 1) & 0xaa) >> 1) | (((((0) + 16) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((0) + 16) + 4) & 0xaa) >> 1) | ((((0) + 16) + 4) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((0) + 16) + 4) & 0xaa) >> 1) | ((((0) + 16) + 4) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((0) + 16) + 4) & 0xaa) >> 1) | ((((0) + 16) + 4) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((0) + 16) + 4) & 0xaa) >> 1) | ((((0) + 16) + 4) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((((0) + 16) + 4) + 1) & 0xaa) >> 1) | (((((0) + 16) + 4) + 1) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((0) + 16) + 4) + 1) & 0xaa) >> 1) | (((((0) + 16) + 4) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((((0) + 16) + 4) + 1) & 0xaa) >> 1) | (((((0) + 16) + 4) + 1) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((0) + 16) + 4) + 1) & 0xaa) >> 1) | (((((0) + 16) + 4) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((((0) + 16) + 4) + 2) & 0xaa) >> 1) | (((((0) + 16) + 4) + 2) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((0) + 16) + 4) + 2) & 0xaa) >> 1) | (((((0) + 16) + 4) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((((0) + 16) + 4) + 2) & 0xaa) >> 1) | (((((0) + 16) + 4) + 2) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((0) + 16) + 4) + 2) & 0xaa) >> 1) | (((((0) + 16) + 4) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((((0) + 16) + 4) + 2) + 1) & 0xaa) >> 1) | ((((((0) + 16) + 4) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((0) + 16) + 4) + 2) + 1) & 0xaa) >> 1) | ((((((0) + 16) + 4) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((0) + 16) + 4) + 2) + 1) & 0xaa) >> 1) | ((((((0) + 16) + 4) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((0) + 16) + 4) + 2) + 1) & 0xaa) >> 1) | ((((((0) + 16) + 4) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((0) + 16) + 8) & 0xaa) >> 1) | ((((0) + 16) + 8) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((0) + 16) + 8) & 0xaa) >> 1) | ((((0) + 16) + 8) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((0) + 16) + 8) & 0xaa) >> 1) | ((((0) + 16) + 8) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((0) + 16) + 8) & 0xaa) >> 1) | ((((0) + 16) + 8) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((((0) + 16) + 8) + 1) & 0xaa) >> 1) | (((((0) + 16) + 8) + 1) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((0) +

16) + 8) + 1) & 0xaa) >> 1) | (((((((0) + 16) + 8) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((0) + 16) + 8) + 1) & 0xaa) >> 1) | (((((((0) + 16) + 8) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((0) + 16) + 8) + 1) & 0xaa) >> 1) | ((((((0) + 16) + 8) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((((0) + 16) + 8) + 2) & 0xaa) >> 1) | (((((((0) + 16) + 8) + 2) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((0) + 16) + 8) + 2) & 0xaa) >> 1) | ((((((0) + 16) + 8) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((0) + 16) + 8) + 2) & 0xaa) >> 1) | (((((((0) + 16) + 8) + 2) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((0) + 16) + 8) + 2) & 0xaa) >> 1) | ((((((0) + 16) + 8) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((((((0) + 16) + 8) + 2) + 1) & 0xaa) >> 1) | (((((((0) + 16) + 8) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((0) + 16) + 8) + 2) + 1) & 0xaa) >> 1) | (((((((0) + 16) + 8) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((((0) + 16) + 8) + 2) + 1) & 0xaa) >> 1) | (((((((0) + 16) + 8) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((0) + 16) + 8) + 2) + 1) & 0xaa) >> 1) | (((((((0) + 16) + 8) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((((((0) + 16) + 8) + 4) & 0xaa) >> 1) | ((((((0) + 16) + 8) + 4) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((0) + 16) + 8) + 4) & 0xaa) >> 1) | ((((((0) + 16) + 8) + 4) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((((((0) + 16) + 8) + 4) & 0xaa) >> 1) | ((((((0) + 16) + 8) + 4) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((0) + 16) + 8) + 4) & 0xaa) >> 1) | ((((((0) + 16) + 8) + 4) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((((((0) + 16) + 8) + 4) + 1) & 0xaa) >> 1) | (((((((0) + 16) + 8) + 4) + 1) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((0) + 16) + 8) + 4) + 1) & 0xaa) >> 1) | (((((((0) + 16) + 8) + 4) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((((0) + 16) + 8) + 4) + 1) & 0xaa) >> 1) | (((((((0) + 16) + 8) + 4) + 1) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((0) + 16) + 8) + 4) + 1) & 0xaa) >> 1) | (((((((0) + 16) + 8) + 4) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((((((0) + 16) + 8) + 4) + 2) & 0xaa) >> 1) | (((((((0) + 16) + 8) + 4) + 2) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((0) + 16) + 8) + 4) + 2) & 0xaa) >> 1) | (((((((0) + 16) + 8) + 4) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((((0) + 16) + 8) + 4) + 2) & 0xaa) >> 1) | (((((((0) + 16) + 8) + 4) + 2) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((0) + 16) + 8) + 4) + 2) & 0xaa) >> 1) | (((((((0) + 16) + 8) + 4) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((((((0) + 16) + 8) + 4) + 2) + 1) & 0xaa) >> 1) | ((((((((0) + 16) + 8) + 4) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((0) + 16) + 8) + 4) + 2) + 1) & 0xaa) >> 1) | ((((((((0) + 16) + 8) + 4) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((((((((0) + 16) + 8) + 4) + 2) + 1) & 0xaa) >> 1) | ((((((((0) + 16) + 8) + 4) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((0) + 16) + 8) + 4) + 2) + 1) & 0xaa) >> 1) | ((((((((0) + 16) + 8) + 4) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((0) + 32) & 0xaa) >> 1) | ((((0) + 32) & 0x55) << 1)) & 0xcc) >> 2) | (((((((0) + 32) & 0xaa) >> 1) | (((0) + 32) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((0) + 32) & 0xaa) >> 1) | (((0) + 32) & 0x55) << 1)) & 0xcc) >> 2) | (((((((0) + 32) & 0xaa) >> 1) | (((0) + 32) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((0) + 32) + 1) & 0xaa) >> 1) | ((((0) + 32) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((0) + 32) + 1) & 0xaa) >> 1) | ((((0) + 32) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((((0) + 32) + 1) & 0xaa) >> 1) | ((((0) + 32) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((0) + 32) + 1) & 0xaa) >> 1) | ((((0) + 32) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((0) + 32) + 2) & 0xaa) >> 1) | ((((0) + 32) + 2) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((0) + 32) + 2) & 0xaa) >> 1) | ((((0) + 32) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((((0) + 32) + 2) & 0xaa) >> 1) | ((((0) + 32) + 2) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((0) + 32) + 2) & 0xaa) >> 1) | ((((0) + 32) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((((0) + 32) + 2) + 1) & 0xaa) >> 1) | (((((0) + 32) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((0) + 32) + 2) + 1) & 0xaa) >> 1) | (((((0) + 32) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((0) + 32) + 2) + 1) & 0xaa) >> 1) | (((((0) + 32) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((0) + 32) + 2) + 1) & 0xaa) >> 1) | (((((0) + 32) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((((0) + 32) + 4) & 0xaa) >> 1) | (((((0) + 32) + 4) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((0) + 32) + 4) & 0xaa) >> 1) | (((((0) + 32) + 4) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((0) + 32) + 4) & 0xaa) >> 1) | (((((0) + 32) + 4) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((0) + 32) + 4) & 0xaa) >> 1) | (((((0) + 32) + 4) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((((0) + 32) + 4) + 1) & 0xaa) >> 1) | ((((((0) + 32) + 4) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((0) + 32) + 4) + 1) & 0xaa) >> 1) | ((((((0) + 32) + 4) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((((((0) + 32) + 4) + 1) & 0xaa) >> 1) | ((((((0) + 32) + 4) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((0) + 32) + 4) + 1) & 0xaa) >> 1) | ((((((0) + 32) + 4) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((((0) + 32) + 4) + 2) & 0xaa) >> 1) | ((((((0) + 32) + 4) + 2) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((0) + 32) + 4) + 2) & 0xaa) >> 1) | ((((((0) + 32) + 4) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((((((0) + 32) + 4) + 2) & 0xaa) >> 1) | ((((((0) + 32) + 4) + 2) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((0) + 32) + 4) + 2) &

0xaa) >> 1) | ((((((0) + 32) + 4) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((((0) + 32) + 4) + 2) + 1) & 0xaa) >> 1) | (((((((0) + 32) + 4) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((0) + 32) + 4) + 2) + 1) & 0xaa) >> 1) | ((((((0) + 32) + 4) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((((0) + 32) + 4) + 2) + 1) & 0xaa) >> 1) | (((((((0) + 32) + 4) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((0) + 32) + 4) + 2) + 1) & 0xaa) >> 1) | (((((((0) + 32) + 4) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((0) + 32) + 8) & 0xaa) >> 1) | ((((0) + 32) + 8) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((0) + 32) + 8) & 0xaa) >> 1) | ((((0) + 32) + 8) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((((0) + 32) + 8) & 0xaa) >> 1) | ((((0) + 32) + 8) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((0) + 32) + 8) & 0xaa) >> 1) | ((((0) + 32) + 8) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((((0) + 32) + 8) + 1) & 0xaa) >> 1) | (((((0) + 32) + 8) + 1) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((0) + 32) + 8) + 1) & 0xaa) >> 1) | ((((((0) + 32) + 8) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((((((0) + 32) + 8) + 1) & 0xaa) >> 1) | ((((((0) + 32) + 8) + 1) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((0) + 32) + 8) + 1) & 0xaa) >> 1) | ((((((0) + 32) + 8) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((((0) + 32) + 8) + 2) & 0xaa) >> 1) | ((((((0) + 32) + 8) + 2) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((0) + 32) + 8) + 2) & 0xaa) >> 1) | ((((((0) + 32) + 8) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((((((0) + 32) + 8) + 2) & 0xaa) >> 1) | ((((((0) + 32) + 8) + 2) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((0) + 32) + 8) + 2) & 0xaa) >> 1) | ((((((0) + 32) + 8) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((((0) + 32) + 8) + 2) + 1) & 0xaa) >> 1) | (((((((0) + 32) + 8) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((0) + 32) + 8) + 2) + 1) & 0xaa) >> 1) | (((((((0) + 32) + 8) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((((0) + 32) + 8) + 2) + 1) & 0xaa) >> 1) | (((((((0) + 32) + 8) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((0) + 32) + 8) + 2) + 1) & 0xaa) >> 1) | (((((((0) + 32) + 8) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((((0) + 32) + 8) + 4) + 1) & 0xaa) >> 1) | (((((((0) + 32) + 8) + 4) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((0) + 32) + 8) + 4) + 1) & 0xaa) >> 1) | (((((((0) + 32) + 8) + 4) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((((0) + 32) + 8) + 4) + 1) & 0xaa) >> 1) | (((((((0) + 32) + 8) + 4) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((0) + 32) + 8) + 4) + 1) & 0xaa) >> 1) | (((((((0) + 32) + 8) + 4) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((((0) + 32) + 8) + 4) + 2) & 0xaa) >> 1) | (((((((0) + 32) + 8) + 4) + 2) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((0) + 32) + 8) + 4) + 2) & 0xaa) >> 1) | (((((((0) + 32) + 8) + 4) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((((0) + 32) + 8) + 4) + 2) & 0xaa) >> 1) | (((((((0) + 32) + 8) + 4) + 2) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((0) + 32) + 8) + 4) + 2) & 0xaa) >> 1) | (((((((0) + 32) + 8) + 4) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((((((0) + 32) + 8) + 4) + 2) + 1) & 0xaa) >> 1) | ((((((((0) + 32) + 8) + 4) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((0) + 32) + 8) + 4) + 2) + 1) & 0xaa) >> 1) | ((((((((0) + 32) + 8) + 4) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((((((((0) + 32) + 8) + 4) + 2) + 1) & 0xaa) >> 1) | (((((((0) + 32) + 8) + 4) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((0) + 32) + 8) + 4) + 2) + 1) & 0xaa) >> 1) | (((((((0) + 32) + 8) + 4) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((0) + 32) + 16) & 0xaa) >> 1) | ((((0) + 32) + 16) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((0) + 32) + 16) & 0xaa) >> 1) | ((((0) + 32) + 16) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((((0) + 32) + 16) & 0xaa) >> 1) | ((((0) + 32) + 16) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((0) + 32) + 16) & 0xaa) >> 1) | ((((0) + 32) + 16) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((((0) + 32) + 16) + 1) & 0xaa) >> 1) | (((((0) + 32) + 16) + 1) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((0) + 32) + 16) + 1) & 0xaa) >> 1) | (((((0) + 32) + 16) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((((((0) + 32) + 16) + 1) & 0xaa) >> 1) | (((((0) + 32) + 16) + 1) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((0) + 32) + 16) + 1) & 0xaa) >> 1) | (((((0) + 32) + 16) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((((0) + 32) + 16) + 2) & 0xaa) >> 1) | (((((0) + 32) + 16) + 2) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((0) + 32) + 16) + 2) & 0xaa) >> 1) | (((((0) + 32) + 16) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((((((0) + 32) + 16) + 2) & 0xaa) >> 1) | (((((0) + 32) + 16) + 2) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((0) + 32) + 16) + 2) & 0xaa) >> 1) | (((((0) + 32) + 16) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((((0) + 32) + 16) + 2) + 1) & 0xaa) >> 1) | (((((((0) + 32) + 16) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((0) + 32) + 16) + 2) + 1) & 0xaa) >> 1) | (((((((0) + 32) + 16) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((((0) + 32) + 16) + 2) + 1) & 0xaa) >> 1) | (((((((0) + 32) + 16) + 2) + 1) & 0x55) << 1)) & 0xcc)

>> 2) | (((((((((((0) + 32) + 16) + 2) + 1) & 0xaa) >> 1) | (((((((0) + 32) + 16) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((((0) + 32) + 16) + 4) & 0xaa) >> 1) | ((((((0) + 32) + 16) + 4) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((0) + 32) + 16) + 4) & 0xaa) >> 1) | ((((((0) + 32) + 16) + 4) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((0) + 32) + 16) + 4) & 0xaa) >> 1) | ((((((0) + 32) + 16) + 4) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((0) + 32) + 16) + 4) & 0xaa) >> 1) | ((((((0) + 32) + 16) + 4) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((((((0) + 32) + 16) + 4) + 1) & 0xaa) >> 1) | (((((((0) + 32) + 16) + 4) + 1) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((0) + 32) + 16) + 4) + 1) & 0xaa) >> 1) | (((((((0) + 32) + 16) + 4) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((((((0) + 32) + 16) + 4) + 1) & 0xaa) >> 1) | (((((((0) + 32) + 16) + 4) + 1) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((0) + 32) + 16) + 4) + 1) & 0xaa) >> 1) | (((((((0) + 32) + 16) + 4) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((((((0) + 32) + 16) + 4) + 2) & 0xaa) >> 1) | (((((((0) + 32) + 16) + 4) + 2) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((0) + 32) + 16) + 4) + 2) & 0xaa) >> 1) | (((((((0) + 32) + 16) + 4) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((((((0) + 32) + 16) + 4) + 2) & 0xaa) >> 1) | (((((((0) + 32) + 16) + 4) + 2) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((0) + 32) + 16) + 4) + 2) & 0xaa) >> 1) | (((((((0) + 32) + 16) + 4) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((((((0) + 32) + 16) + 4) + 2) + 1) & 0xaa) >> 1) | ((((((((0) + 32) + 16) + 4) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((0) + 32) + 16) + 4) + 2) + 1) & 0xaa) >> 1) | ((((((((0) + 32) + 16) + 4) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((((0) + 32) + 16) + 4) + 2) + 1) & 0xaa) >> 1) | ((((((((0) + 32) + 16) + 4) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((0) + 32) + 16) + 4) + 2) + 1) & 0xaa) >> 1) | ((((((((0) + 32) + 16) + 4) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((((((0) + 32) + 16) + 8) & 0xaa) >> 1) | ((((((0) + 32) + 16) + 8) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((0) + 32) + 16) + 8) & 0xaa) >> 1) | ((((((0) + 32) + 16) + 8) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((0) + 32) + 16) + 8) & 0xaa) >> 1) | ((((((0) + 32) + 16) + 8) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((0) + 32) + 16) + 8) & 0xaa) >> 1) | ((((((0) + 32) + 16) + 8) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((((((0) + 32) + 16) + 8) + 1) & 0xaa) >> 1) | (((((((0) + 32) + 16) + 8) + 1) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((0) + 32) + 16) + 8) + 1) & 0xaa) >> 1) | (((((((0) + 32) + 16) + 8) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((((((0) + 32) + 16) + 8) + 1) & 0xaa) >> 1) | (((((((0) + 32) + 16) + 8) + 1) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((0) + 32) + 16) + 8) + 1) & 0xaa) >> 1) | (((((((0) + 32) + 16) + 8) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((((((0) + 32) + 16) + 8) + 2) & 0xaa) >> 1) | (((((((0) + 32) + 16) + 8) + 2) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((0) + 32) + 16) + 8) + 2) & 0xaa) >> 1) | (((((((0) + 32) + 16) + 8) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((((((0) + 32) + 16) + 8) + 2) & 0xaa) >> 1) | (((((((0) + 32) + 16) + 8) + 2) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((0) + 32) + 16) + 8) + 2) & 0xaa) >> 1) | (((((((0) + 32) + 16) + 8) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((((((0) + 32) + 16) + 8) + 2) + 1) & 0xaa) >> 1) | ((((((((0) + 32) + 16) + 8) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((0) + 32) + 16) + 8) + 2) + 1) & 0xaa) >> 1) | ((((((((0) + 32) + 16) + 8) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((((0) + 32) + 16) + 8) + 2) + 1) & 0xaa) >> 1) | ((((((((0) + 32) + 16) + 8) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((0) + 32) + 16) + 8) + 2) + 1) & 0xaa) >> 1) | ((((((((0) + 32) + 16) + 8) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((((((0) + 32) + 16) + 8) + 4) & 0xaa) >> 1) | (((((((0) + 32) + 16) + 8) + 4) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((0) + 32) + 16) + 8) + 4) & 0xaa) >> 1) | (((((((0) + 32) + 16) + 8) + 4) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((((((0) + 32) + 16) + 8) + 4) & 0xaa) >> 1) | (((((((0) + 32) + 16) + 8) + 4) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((0) + 32) + 16) + 8) + 4) & 0xaa) >> 1) | (((((((0) + 32) + 16) + 8) + 4) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((((((0) + 32) + 16) + 8) + 4) + 1) & 0xaa) >> 1) | ((((((((0) + 32) + 16) + 8) + 4) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((0) + 32) + 16) + 8) + 4) + 1) & 0xaa) >> 1) | ((((((((0) + 32) + 16) + 8) + 4) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((((0) + 32) + 16) + 8) + 4) + 1) & 0xaa) >> 1) | ((((((((0) + 32) + 16) + 8) + 4) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((0) + 32) + 16) + 8) + 4) + 1) & 0xaa) >> 1) | ((((((((0) + 32) + 16) + 8) + 4) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((((((0) + 32) + 16) + 8) + 4) + 2) & 0xaa) >> 1) | ((((((((0) + 32) + 16) + 8) + 4) + 2) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((0) + 32) + 16) + 8) + 4) + 2) & 0xaa) >> 1) | ((((((((0) + 32) + 16) + 8) + 4) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((((0) + 32) + 16) + 8) + 4) + 2) & 0xaa) >> 1) | ((((((((0) + 32) + 16) + 8) + 4) + 2) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((0) + 32) + 16) + 8) + 4) + 2) & 0xaa) >> 1) | ((((((((0) + 32) + 16) + 8) + 4) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((((((0) + 32) + 16) + 8) + 4) + 2) + 1) & 0xaa) >> 1) | (((((((((0) + 32) + 16) + 8) + 4) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((((0) + 32) + 16) + 8) + 4) + 2) + 1) & 0xaa) >> 1) | ((((

(((((0) + 32) + 16) + 8) + 4) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((((((((0) + 32) + 16) + 8) + 4) + 2) + 1) & 0xaa) >> 1) | (((((((((0) + 32) + 16) + 8) + 4) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((((0) + 32) + 16) + 8) + 4) + 2) + 1) & 0xaa) >> 1) | (((((((((0) + 32) + 16) + 8) + 4) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((0) + 64) & 0xaa) >> 1) | ((((0) + 64) & 0x55) << 1)) & 0xcc) >> 2) | (((((((0) + 64) & 0xaa) >> 1) | ((((0) + 64) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((0) + 64) & 0xaa) >> 1) | (((0) + 64) & 0x55) << 1)) & 0xcc) >> 2) | (((((((0) + 64) & 0xaa) >> 1) | (((0) + 64) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((0) + 64) + 1) & 0xaa) >> 1) | (((((0) + 64) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((0) + 64) + 1) & 0xaa) >> 1) | (((((0) + 64) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((0) + 64) + 1) & 0xaa) >> 1) | (((((0) + 64) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((0) + 64) + 1) & 0xaa) >> 1) | (((((0) + 64) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((0) + 64) + 2) & 0xaa) >> 1) | (((((0) + 64) + 2) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((0) + 64) + 2) & 0xaa) >> 1) | (((((0) + 64) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((0) + 64) + 2) & 0xaa) >> 1) | (((((0) + 64) + 2) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((0) + 64) + 2) & 0xaa) >> 1) | (((((0) + 64) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((((0) + 64) + 2) + 1) & 0xaa) >> 1) | ((((((0) + 64) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((0) + 64) + 2) + 1) & 0xaa) >> 1) | ((((((0) + 64) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((0) + 64) + 2) + 1) & 0xaa) >> 1) | ((((((0) + 64) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((0) + 64) + 2) + 1) & 0xaa) >> 1) | (((((((0) + 64) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((((0) + 64) + 4) & 0xaa) >> 1) | (((((0) + 64) + 4) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((0) + 64) + 4) & 0xaa) >> 1) | (((((0) + 64) + 4) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((((0) + 64) + 4) & 0xaa) >> 1) | (((((0) + 64) + 4) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((0) + 64) + 4) & 0xaa) >> 1) | (((((0) + 64) + 4) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((((0) + 64) + 4) + 1) & 0xaa) >> 1) | ((((((0) + 64) + 4) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((0) + 64) + 4) + 1) & 0xaa) >> 1) | ((((((0) + 64) + 4) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((0) + 64) + 4) + 1) & 0xaa) >> 1) | ((((((0) + 64) + 4) + 1) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((0) + 64) + 4) + 1) & 0xaa) >> 1) | (((((((0) + 64) + 4) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((((0) + 64) + 4) + 2) & 0xaa) >> 1) | ((((((0) + 64) + 4) + 2) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((0) + 64) + 4) + 2) & 0xaa) >> 1) | ((((((0) + 64) + 4) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((0) + 64) + 4) + 2) & 0xaa) >> 1) | ((((((0) + 64) + 4) + 2) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((0) + 64) + 4) + 2) & 0xaa) >> 1) | (((((((0) + 64) + 4) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((((((0) + 64) + 4) + 2) + 1) & 0xaa) >> 1) | (((((((0) + 64) + 4) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((0) + 64) + 4) + 2) + 1) & 0xaa) >> 1) | ((((((((0) + 64) + 4) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((((((0) + 64) + 4) + 2) + 1) & 0xaa) >> 1) | (((((((0) + 64) + 4) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((0) + 64) + 4) + 2) + 1) & 0xaa) >> 1) | ((((((((0) + 64) + 4) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((((0) + 64) + 8) & 0xaa) >> 1) | ((((0) + 64) + 8) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((0) + 64) + 8) & 0xaa) >> 1) | ((((0) + 64) + 8) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((0) + 64) + 8) & 0xaa) >> 1) | ((((0) + 64) + 8) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((0) + 64) + 8) & 0xaa) >> 1) | ((((0) + 64) + 8) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((((0) + 64) + 8) + 1) & 0xaa) >> 1) | (((((0) + 64) + 8) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((0) + 64) + 8) + 1) & 0xaa) >> 1) | ((((((0) + 64) + 8) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((0) + 64) + 8) + 1) & 0xaa) >> 1) | ((((((0) + 64) + 8) + 1) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((0) + 64) + 8) + 1) & 0xaa) >> 1) | ((((((0) + 64) + 8) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((((0) + 64) + 8) + 2) & 0xaa) >> 1) | ((((((0) + 64) + 8) + 2) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((0) + 64) + 8) + 2) & 0xaa) >> 1) | ((((((0) + 64) + 8) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((0) + 64) + 8) + 2) & 0xaa) >> 1) | ((((((0) + 64) + 8) + 2) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((0) + 64) + 8) + 2) & 0xaa) >> 1) | ((((((0) + 64) + 8) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((((((0) + 64) + 8) + 2) + 1) & 0xaa) >> 1) | (((((((0) + 64) + 8) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((0) + 64) + 8) + 2) + 1) & 0xaa) >> 1) | ((((((((0) + 64) + 8) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((((((0) + 64) + 8) + 2) + 1) & 0xaa) >> 1) | (((((((0) + 64) + 8) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((0) + 64) + 8) + 2) + 1) & 0xaa) >> 1) | ((((((((0) + 64) + 8) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((((0) + 64) + 8) + 4) & 0xaa) >> 1) | (((((0) + 64) + 8) + 4) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((0) + 64) + 8) + 4) & 0xaa) >> 1) | (((((0) + 64) + 8) + 4) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((((0) + 64) + 8) + 4) & 0xaa) >> 1) | (((((0) + 64) + 8) + 4) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((0) + 64) + 8) + 4) & 0xaa) >> 1) | ((((((0) + 64) + 8) + 4) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((((((0) + 64) + 8) + 4) + 1) & 0xaa) >> 1)

| (((((((0) + 64) + 8) + 4) + 1) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((0) + 64) + 8) + 4) + 1) & 0xaa) >> 1) | (((((((0) + 64) + 8) + 4) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((0) + 64) + 8) + 4) + 1) & 0xaa) >> 1) | (((((((0) + 64) + 8) + 4) + 1) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((0) + 64) + 8) + 4) + 1) & 0xaa) >> 1) | (((((((0) + 64) + 8) + 4) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((((((0) + 64) + 8) + 4) + 2) & 0xaa) >> 1) | (((((((0) + 64) + 8) + 4) + 2) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((0) + 64) + 8) + 4) + 2) & 0xaa) >> 1) | (((((((0) + 64) + 8) + 4) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((((((0) + 64) + 8) + 4) + 2) & 0xaa) >> 1) | (((((((0) + 64) + 8) + 4) + 2) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((0) + 64) + 8) + 4) + 2) & 0xaa) >> 1) | (((((((0) + 64) + 8) + 4) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((((((0) + 64) + 8) + 4) + 2) + 1) & 0xaa) >> 1) | ((((((((0) + 64) + 8) + 4) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((0) + 64) + 8) + 4) + 2) + 1) & 0xaa) >> 1) | ((((((((0) + 64) + 8) + 4) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((((0) + 64) + 8) + 4) + 2) + 1) & 0xaa) >> 1) | ((((((((0) + 64) + 8) + 4) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((0) + 64) + 8) + 4) + 2) + 1) & 0xaa) >> 1) | ((((((((0) + 64) + 8) + 4) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((((0) + 64) + 16) & 0xaa) >> 1) | (((((0) + 64) + 16) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((0) + 64) + 16) & 0xaa) >> 1) | (((((0) + 64) + 16) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((((0) + 64) + 16) & 0xaa) >> 1) | (((((0) + 64) + 16) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((0) + 64) + 16) & 0xaa) >> 1) | (((((0) + 64) + 16) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((((0) + 64) + 16) + 1) & 0xaa) >> 1) | ((((((0) + 64) + 16) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((0) + 64) + 16) + 1) & 0xaa) >> 1) | ((((((0) + 64) + 16) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((0) + 64) + 16) + 1) & 0xaa) >> 1) | ((((((0) + 64) + 16) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((0) + 64) + 16) + 1) & 0xaa) >> 1) | ((((((0) + 64) + 16) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((((0) + 64) + 16) + 2) & 0xaa) >> 1) | ((((((0) + 64) + 16) + 2) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((0) + 64) + 16) + 2) & 0xaa) >> 1) | ((((((0) + 64) + 16) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((0) + 64) + 16) + 2) & 0xaa) >> 1) | ((((((0) + 64) + 16) + 2) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((0) + 64) + 16) + 2) & 0xaa) >> 1) | ((((((0) + 64) + 16) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((((((0) + 64) + 16) + 2) + 1) & 0xaa) >> 1) | (((((((0) + 64) + 16) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((0) + 64) + 16) + 2) + 1) & 0xaa) >> 1) | (((((((0) + 64) + 16) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((((((0) + 64) + 16) + 2) + 1) & 0xaa) >> 1) | (((((((0) + 64) + 16) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((0) + 64) + 16) + 2) + 1) & 0xaa) >> 1) | (((((((0) + 64) + 16) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((((0) + 64) + 16) + 4) & 0xaa) >> 1) | ((((((0) + 64) + 16) + 4) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((0) + 64) + 16) + 4) & 0xaa) >> 1) | ((((((0) + 64) + 16) + 4) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((0) + 64) + 16) + 4) & 0xaa) >> 1) | ((((((0) + 64) + 16) + 4) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((0) + 64) + 16) + 4) & 0xaa) >> 1) | ((((((0) + 64) + 16) + 4) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((((((0) + 64) + 16) + 4) + 1) & 0xaa) >> 1) | (((((((0) + 64) + 16) + 4) + 1) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((0) + 64) + 16) + 4) + 1) & 0xaa) >> 1) | (((((((0) + 64) + 16) + 4) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((((((0) + 64) + 16) + 4) + 1) & 0xaa) >> 1) | (((((((0) + 64) + 16) + 4) + 1) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((0) + 64) + 16) + 4) + 1) & 0xaa) >> 1) | (((((((0) + 64) + 16) + 4) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((((((0) + 64) + 16) + 4) + 2) & 0xaa) >> 1) | (((((((0) + 64) + 16) + 4) + 2) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((0) + 64) + 16) + 4) + 2) & 0xaa) >> 1) | (((((((0) + 64) + 16) + 4) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((((((0) + 64) + 16) + 4) + 2) & 0xaa) >> 1) | (((((((0) + 64) + 16) + 4) + 2) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((0) + 64) + 16) + 4) + 2) & 0xaa) >> 1) | (((((((0) + 64) + 16) + 4) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((((((0) + 64) + 16) + 4) + 2) + 1) & 0xaa) >> 1) | ((((((((0) + 64) + 16) + 4) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((0) + 64) + 16) + 4) + 2) + 1) & 0xaa) >> 1) | ((((((((0) + 64) + 16) + 4) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((((0) + 64) + 16) + 4) + 2) + 1) & 0xaa) >> 1) | ((((((((0) + 64) + 16) + 4) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((0) + 64) + 16) + 4) + 2) + 1) & 0xaa) >> 1) | ((((((((0) + 64) + 16) + 4) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((((0) + 64) + 16) + 8) & 0xaa) >> 1) | ((((((0) + 64) + 16) + 8) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((0) + 64) + 16) + 8) & 0xaa) >> 1) | ((((((0) + 64) + 16) + 8) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((0) + 64) + 16) + 8) & 0xaa) >> 1) | ((((((0) + 64) + 16) + 8) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((0) + 64) + 16) + 8) & 0xaa) >> 1) | ((((((0) + 64) + 16) + 8) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((((((0) + 64) + 16) + 8) + 1) & 0xaa) >> 1) | (((((((0) + 64) + 16) + 8) + 1) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((0) + 64) + 16) + 8) + 1) & 0xaa) >> 1) | (((((((0) + 64) + 16) + 8) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((((

((0) + 64) + 16) + 8) + 1) & 0xaa) >> 1) | ((((((((0) + 64) + 16) + 8) + 1) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((0) + 64) + 16) + 8) + 1) & 0xaa) >> 1) | (((((((((0) + 64) + 16) + 8) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((((((0) + 64) + 16) + 8) + 2) & 0xaa) >> 1) | (((((((0) + 64) + 16) + 8) + 2) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((0) + 64) + 16) + 8) + 2) & 0xaa) >> 1) | (((((((0) + 64) + 16) + 8) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((((((0) + 64) + 16) + 8) + 2) & 0xaa) >> 1) | (((((((0) + 64) + 16) + 8) + 2) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((0) + 64) + 16) + 8) + 2) & 0xaa) >> 1) | (((((((0) + 64) + 16) + 8) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((((((0) + 64) + 16) + 8) + 2) + 1) & 0xaa) >> 1) | ((((((((0) + 64) + 16) + 8) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((0) + 64) + 16) + 8) + 2) + 1) & 0xaa) >> 1) | ((((((((0) + 64) + 16) + 8) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((((0) + 64) + 16) + 8) + 2) + 1) & 0xaa) >> 1) | ((((((((0) + 64) + 16) + 8) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((0) + 64) + 16) + 8) + 2) + 1) & 0xaa) >> 1) | ((((((((0) + 64) + 16) + 8) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((((((0) + 64) + 16) + 8) + 4) & 0xaa) >> 1) | (((((((0) + 64) + 16) + 8) + 4) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((0) + 64) + 16) + 8) + 4) & 0xaa) >> 1) | (((((((0) + 64) + 16) + 8) + 4) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((((((0) + 64) + 16) + 8) + 4) & 0xaa) >> 1) | (((((((0) + 64) + 16) + 8) + 4) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((0) + 64) + 16) + 8) + 4) & 0xaa) >> 1) | (((((((0) + 64) + 16) + 8) + 4) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((((((0) + 64) + 16) + 8) + 4) + 1) & 0xaa) >> 1) | ((((((((0) + 64) + 16) + 8) + 4) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((0) + 64) + 16) + 8) + 4) + 1) & 0xaa) >> 1) | ((((((((0) + 64) + 16) + 8) + 4) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((((0) + 64) + 16) + 8) + 4) + 1) & 0xaa) >> 1) | ((((((((0) + 64) + 16) + 8) + 4) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((0) + 64) + 16) + 8) + 4) + 1) & 0xaa) >> 1) | ((((((((0) + 64) + 16) + 8) + 4) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((((((((0) + 64) + 16) + 8) + 4) + 2) + 1) & 0xaa) >> 1) | (((((((((0) + 64) + 16) + 8) + 4) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((((0) + 64) + 16) + 8) + 4) + 2) + 1) & 0xaa) >> 1) | (((((((((0) + 64) + 16) + 8) + 4) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((((((((0) + 64) + 16) + 8) + 4) + 2) + 1) & 0xaa) >> 1) | (((((((((0) + 64) + 16) + 8) + 4) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((((0) + 64) + 16) + 8) + 4) + 2) + 1) & 0xaa) >> 1) | (((((((((0) + 64) + 16) + 8) + 4) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((((((((0) + 64) + 32) & 0xaa) >> 1) | (((((((0) + 64) + 32) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((0) + 64) + 32) & 0xaa) >> 1) | (((((((0) + 64) + 32) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((((((0) + 64) + 32) & 0xaa) >> 1) | (((((((0) + 64) + 32) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((0) + 64) + 32) & 0xaa) >> 1) | (((((((0) + 64) + 32) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((((((0) + 64) + 32) + 1) & 0xaa) >> 1) | (((((((0) + 64) + 32) + 1) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((0) + 64) + 32) + 1) & 0xaa) >> 1) | (((((((0) + 64) + 32) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((((((0) + 64) + 32) + 1) & 0xaa) >> 1) | (((((((0) + 64) + 32) + 1) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((0) + 64) + 32) + 1) & 0xaa) >> 1) | (((((((0) + 64) + 32) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((((((0) + 64) + 32) + 2) & 0xaa) >> 1) | (((((((0) + 64) + 32) + 2) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((0) + 64) + 32) + 2) & 0xaa) >> 1) | (((((((0) + 64) + 32) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((((((0) + 64) + 32) + 2) & 0xaa) >> 1) | (((((((0) + 64) + 32) + 2) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((0) + 64) + 32) + 2) & 0xaa) >> 1) | (((((((0) + 64) + 32) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((((((0) + 64) + 32) + 2) + 1) & 0xaa) >> 1) | ((((((((0) + 64) + 32) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((0) + 64) + 32) + 2) + 1) & 0xaa) >> 1) | ((((((((0) + 64) + 32) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((((0) + 64) + 32) + 2) + 1) & 0xaa) >> 1) | ((((((((0) + 64) + 32) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((0) + 64) + 32) + 2) + 1) & 0xaa) >> 1) | ((((((((0) + 64) + 32) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((((((0) + 64) + 32) + 4) & 0xaa) >> 1) | (((((((0) + 64) + 32) + 4) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((0) + 64) + 32) + 4) & 0xaa) >> 1) | (((((((0) + 64) + 32) + 4) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((((((0) + 64) + 32) + 4) & 0xaa) >> 1) | (((((((0) + 64) + 32) + 4) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((0) + 64) + 32) + 4) & 0xaa) >> 1) | (((((((0) + 64) + 32) + 4) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((((((0) + 64) + 32) + 4) + 1) & 0xaa) >> 1) | (((((((0) + 64) + 32) + 4) + 1) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((0) + 64) + 32) + 4) + 1) & 0xaa) >> 1) | (((((((0) + 64) + 32) + 4) + 1) & 0x55) << 1)) & 0x33) << 2))

& 0xf0) >> 4) | (((((((((((((0) + 64) + 32) + 4) + 1) & 0xaa) >> 1) | ((((((((0) + 64) + 32) + 4) + 1) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((0) + 64) + 32) + 4) + 1) & 0xaa) >> 1) | (((((((0) + 64) + 32) + 4) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((((((0) + 64) + 32) + 4) + 2) & 0xaa) >> 1) | (((((((0) + 64) + 32) + 4) + 2) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((0) + 64) + 32) + 4) + 2) & 0xaa) >> 1) | (((((((0) + 64) + 32) + 4) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((((((0) + 64) + 32) + 4) + 2) & 0xaa) >> 1) | (((((((0) + 64) + 32) + 4) + 2) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((0) + 64) + 32) + 4) + 2) & 0xaa) >> 1) | (((((((0) + 64) + 32) + 4) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((((((0) + 64) + 32) + 4) + 2) + 1) & 0xaa) >> 1) | ((((((((0) + 64) + 32) + 4) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((0) + 64) + 32) + 4) + 2) + 1) & 0xaa) >> 1) | ((((((((0) + 64) + 32) + 4) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((((0) + 64) + 32) + 4) + 2) + 1) & 0xaa) >> 1) | ((((((((0) + 64) + 32) + 4) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((0) + 64) + 32) + 4) + 2) + 1) & 0xaa) >> 1) | ((((((((0) + 64) + 32) + 4) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((((0) + 64) + 32) + 8) & 0xaa) >> 1) | ((((((0) + 64) + 32) + 8) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((0) + 64) + 32) + 8) & 0xaa) >> 1) | ((((((0) + 64) + 32) + 8) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((0) + 64) + 32) + 8) & 0xaa) >> 1) | ((((((0) + 64) + 32) + 8) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((0) + 64) + 32) + 8) & 0xaa) >> 1) | ((((((0) + 64) + 32) + 8) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((((((0) + 64) + 32) + 8) + 1) & 0xaa) >> 1) | (((((((0) + 64) + 32) + 8) + 1) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((0) + 64) + 32) + 8) + 1) & 0xaa) >> 1) | (((((((0) + 64) + 32) + 8) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((((((0) + 64) + 32) + 8) + 1) & 0xaa) >> 1) | (((((((0) + 64) + 32) + 8) + 1) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((0) + 64) + 32) + 8) + 1) & 0xaa) >> 1) | (((((((0) + 64) + 32) + 8) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((((((0) + 64) + 32) + 8) + 2) & 0xaa) >> 1) | (((((((0) + 64) + 32) + 8) + 2) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((0) + 64) + 32) + 8) + 2) & 0xaa) >> 1) | (((((((0) + 64) + 32) + 8) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((((((0) + 64) + 32) + 8) + 2) & 0xaa) >> 1) | (((((((0) + 64) + 32) + 8) + 2) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((0) + 64) + 32) + 8) + 2) & 0xaa) >> 1) | (((((((0) + 64) + 32) + 8) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((((((0) + 64) + 32) + 8) + 2) + 1) & 0xaa) >> 1) | ((((((((0) + 64) + 32) + 8) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((0) + 64) + 32) + 8) + 2) + 1) & 0xaa) >> 1) | ((((((((0) + 64) + 32) + 8) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((((((0) + 64) + 32) + 8) + 2) + 1) & 0xaa) >> 1) | ((((((((0) + 64) + 32) + 8) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((0) + 64) + 32) + 8) + 2) + 1) & 0xaa) >> 1) | ((((((((0) + 64) + 32) + 8) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((((((0) + 64) + 32) + 8) + 4) & 0xaa) >> 1) | (((((((0) + 64) + 32) + 8) + 4) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((0) + 64) + 32) + 8) + 4) & 0xaa) >> 1) | (((((((0) + 64) + 32) + 8) + 4) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((((((0) + 64) + 32) + 8) + 4) & 0xaa) >> 1) | (((((((0) + 64) + 32) + 8) + 4) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((0) + 64) + 32) + 8) + 4) & 0xaa) >> 1) | (((((((0) + 64) + 32) + 8) + 4) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((((((0) + 64) + 32) + 8) + 4) + 1) & 0xaa) >> 1) | ((((((((0) + 64) + 32) + 8) + 4) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((0) + 64) + 32) + 8) + 4) + 1) & 0xaa) >> 1) | ((((((((0) + 64) + 32) + 8) + 4) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((((0) + 64) + 32) + 8) + 4) + 1) & 0xaa) >> 1) | ((((((((0) + 64) + 32) + 8) + 4) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((0) + 64) + 32) + 8) + 4) + 1) & 0xaa) >> 1) | ((((((((0) + 64) + 32) + 8) + 4) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((((((0) + 64) + 32) + 8) + 4) + 2) & 0xaa) >> 1) | ((((((((0) + 64) + 32) + 8) + 4) + 2) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((0) + 64) + 32) + 8) + 4) + 2) & 0xaa) >> 1) | ((((((((0) + 64) + 32) + 8) + 4) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((((0) + 64) + 32) + 8) + 4) + 2) & 0xaa) >> 1) | ((((((((0) + 64) + 32) + 8) + 4) + 2) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((0) + 64) + 32) + 8) + 4) + 2) & 0xaa) >> 1) | ((((((((0) + 64) + 32) + 8) + 4) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((((((((0) + 64) + 32) + 8) + 4) + 2) + 1) & 0xaa) >> 1) | (((((((((0) + 64) + 32) + 8) + 4) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((((0) + 64) + 32) + 8) + 4) + 2) + 1) & 0xaa) >> 1) | (((((((((0) + 64) + 32) + 8) + 4) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((((((((((0) + 64) + 32) + 8) + 4) + 2) + 1) & 0xaa) >> 1) | (((((((((0) + 64) + 32) + 8) + 4) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((((0) + 64) + 32) + 8) + 4) + 2) + 1) & 0xaa) >> 1) | (((((((((0) + 64) + 32) + 8) + 4) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((((0) + 64) + 32) + 16) & 0xaa) >> 1) | ((((((0) + 64) + 32) + 16) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((0) + 64) + 32) + 16) & 0xaa) >> 1) | ((((((0) + 64) + 32) + 16) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((0) + 64) + 32) + 16) & 0xaa) >> 1) | ((((((0) + 64) + 32) + 16) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((0) + 64) + 32) + 16) & 0xaa) >> 1) | ((((((0) + 64) + 32) + 16) & 0x55) << 1)) & 0x33) << 2)) &

0x0f) << 4)), (((((((((((((((0) + 64) + 32) + 16) + 1) & 0xaa) >> 1) | (((((((0) + 64) + 32) + 16) + 1) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((0) + 64) + 32) + 16) + 1) & 0xaa) >> 1) | (((((((0) + 64) + 32) + 16) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((((((0) + 64) + 32) + 16) + 1) & 0xaa) >> 1) | (((((((0) + 64) + 32) + 16) + 1) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((0) + 64) + 32) + 16) + 1) & 0xaa) >> 1) | (((((((0) + 64) + 32) + 16) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((((((0) + 64) + 32) + 16) + 2) & 0xaa) >> 1) | (((((((0) + 64) + 32) + 16) + 2) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((0) + 64) + 32) + 16) + 2) & 0xaa) >> 1) | (((((((0) + 64) + 32) + 16) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((((((0) + 64) + 32) + 16) + 2) & 0xaa) >> 1) | (((((((0) + 64) + 32) + 16) + 2) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((0) + 64) + 32) + 16) + 2) & 0xaa) >> 1) | (((((((0) + 64) + 32) + 16) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((((((0) + 64) + 32) + 16) + 2) + 1) & 0xaa) >> 1) | ((((((((0) + 64) + 32) + 16) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((0) + 64) + 32) + 16) + 2) + 1) & 0xaa) >> 1) | ((((((((0) + 64) + 32) + 16) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((((0) + 64) + 32) + 16) + 2) + 1) & 0xaa) >> 1) | ((((((((0) + 64) + 32) + 16) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((0) + 64) + 32) + 16) + 2) + 1) & 0xaa) >> 1) | ((((((((0) + 64) + 32) + 16) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((((((0) + 64) + 32) + 16) + 4) & 0xaa) >> 1) | (((((((0) + 64) + 32) + 16) + 4) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((0) + 64) + 32) + 16) + 4) & 0xaa) >> 1) | (((((((0) + 64) + 32) + 16) + 4) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((((((0) + 64) + 32) + 16) + 4) & 0xaa) >> 1) | (((((((0) + 64) + 32) + 16) + 4) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((0) + 64) + 32) + 16) + 4) & 0xaa) >> 1) | (((((((0) + 64) + 32) + 16) + 4) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((((((0) + 64) + 32) + 16) + 4) + 1) & 0xaa) >> 1) | ((((((((0) + 64) + 32) + 16) + 4) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((0) + 64) + 32) + 16) + 4) + 1) & 0xaa) >> 1) | ((((((((0) + 64) + 32) + 16) + 4) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((((0) + 64) + 32) + 16) + 4) + 1) & 0xaa) >> 1) | ((((((((0) + 64) + 32) + 16) + 4) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((0) + 64) + 32) + 16) + 4) + 1) & 0xaa) >> 1) | ((((((((0) + 64) + 32) + 16) + 4) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((((((0) + 64) + 32) + 16) + 4) + 2) & 0xaa) >> 1) | ((((((((0) + 64) + 32) + 16) + 4) + 2) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((0) + 64) + 32) + 16) + 4) + 2) & 0xaa) >> 1) | ((((((((0) + 64) + 32) + 16) + 4) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((((0) + 64) + 32) + 16) + 4) + 2) & 0xaa) >> 1) | ((((((((0) + 64) + 32) + 16) + 4) + 2) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((0) + 64) + 32) + 16) + 4) + 2) & 0xaa) >> 1) | ((((((((0) + 64) + 32) + 16) + 4) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((((((((0) + 64) + 32) + 16) + 4) + 2) + 1) & 0xaa) >> 1) | (((((((((0) + 64) + 32) + 16) + 4) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((((0) + 64) + 32) + 16) + 4) + 2) + 1) & 0xaa) >> 1) | (((((((((0) + 64) + 32) + 16) + 4) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((((((((0) + 64) + 32) + 16) + 4) + 2) + 1) & 0xaa) >> 1) | (((((((((0) + 64) + 32) + 16) + 4) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((((0) + 64) + 32) + 16) + 4) + 2) + 1) & 0xaa) >> 1) | (((((((((0) + 64) + 32) + 16) + 4) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((((((0) + 64) + 32) + 16) + 8) & 0xaa) >> 1) | (((((((0) + 64) + 32) + 16) + 8) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((0) + 64) + 32) + 16) + 8) & 0xaa) >> 1) | (((((((0) + 64) + 32) + 16) + 8) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((((((0) + 64) + 32) + 16) + 8) & 0xaa) >> 1) | (((((((0) + 64) + 32) + 16) + 8) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((0) + 64) + 32) + 16) + 8) & 0xaa) >> 1) | (((((((0) + 64) + 32) + 16) + 8) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((((((0) + 64) + 32) + 16) + 8) + 1) & 0xaa) >> 1) | ((((((((0) + 64) + 32) + 16) + 8) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((0) + 64) + 32) + 16) + 8) + 1) & 0xaa) >> 1) | ((((((((0) + 64) + 32) + 16) + 8) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((((0) + 64) + 32) + 16) + 8) + 1) & 0xaa) >> 1) | ((((((((0) + 64) + 32) + 16) + 8) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((0) + 64) + 32) + 16) + 8) + 1) & 0xaa) >> 1) | ((((((((0) + 64) + 32) + 16) + 8) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((((((0) + 64) + 32) + 16) + 8) + 2) & 0xaa) >> 1) | ((((((((0) + 64) + 32) + 16) + 8) + 2) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((0) + 64) + 32) + 16) + 8) + 2) & 0xaa) >> 1) | ((((((((0) + 64) + 32) + 16) + 8) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((((0) + 64) + 32) + 16) + 8) + 2) & 0xaa) >> 1) | ((((((((0) + 64) + 32) + 16) + 8) + 2) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((0) + 64) + 32) + 16) + 8) + 2) & 0xaa) >> 1) | ((((((((0) + 64) + 32) + 16) + 8) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((((((((0) + 64) + 32) + 16) + 8) + 2) + 1) & 0xaa) >> 1) | (((((((((0) + 64) + 32) + 16) + 8) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((((0) + 64) + 32) + 16) + 8) + 2) + 1) & 0xaa) >> 1) | (((((((((0) + 64) + 32) + 16) + 8) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((((((((0) + 64) + 32) + 16) + 8) + 2) + 1) & 0xaa) >> 1) | (((((((((0) + 64) + 32) + 16) + 8) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((((0) + 64) + 32) + 16) + 8) + 2)

+ 1) & 0xaa) >> 1) | (((((((((0) + 64) + 32) + 16) + 8) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((((((0) + 64) + 32) + 16) + 8) + 4) & 0xaa) >> 1) | ((((((((0) + 64) + 32) + 16) + 8) + 4) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((0) + 64) + 32) + 16) + 8) + 4) & 0xaa) >> 1) | ((((((((0) + 64) + 32) + 16) + 8) + 4) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((((0) + 64) + 32) + 16) + 8) + 4) & 0xaa) >> 1) | ((((((((0) + 64) + 32) + 16) + 8) + 4) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((0) + 64) + 32) + 16) + 8) + 4) & 0xaa) >> 1) | ((((((((0) + 64) + 32) + 16) + 8) + 4) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((((((0) + 64) + 32) + 16) + 8) + 4) + 1) & 0xaa) >> 1) | (((((((((0) + 64) + 32) + 16) + 8) + 4) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((0) + 64) + 32) + 16) + 8) + 4) + 1) & 0xaa) >> 1) | (((((((((0) + 64) + 32) + 16) + 8) + 4) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((((((((0) + 64) + 32) + 16) + 8) + 4) + 1) & 0xaa) >> 1) | (((((((((0) + 64) + 32) + 16) + 8) + 4) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((0) + 64) + 32) + 16) + 8) + 4) + 1) & 0xaa) >> 1) | (((((((((0) + 64) + 32) + 16) + 8) + 4) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((((((0) + 64) + 32) + 16) + 8) + 4) + 2) & 0xaa) >> 1) | (((((((((0) + 64) + 32) + 16) + 8) + 4) + 2) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((0) + 64) + 32) + 16) + 8) + 4) + 2) & 0xaa) >> 1) | (((((((((0) + 64) + 32) + 16) + 8) + 4) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((((((((0) + 64) + 32) + 16) + 8) + 4) + 2) & 0xaa) >> 1) | (((((((((0) + 64) + 32) + 16) + 8) + 4) + 2) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((0) + 64) + 32) + 16) + 8) + 4) + 2) & 0xaa) >> 1) | (((((((((0) + 64) + 32) + 16) + 8) + 4) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((((((((0) + 64) + 32) + 16) + 8) + 4) + 2) + 1) & 0xaa) >> 1) | ((((((((((0) + 64) + 32) + 16) + 8) + 4) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((((0) + 64) + 32) + 16) + 8) + 4) + 2) + 1) & 0xaa) >> 1) | ((((((((((0) + 64) + 32) + 16) + 8) + 4) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((((((((((0) + 64) + 32) + 16) + 8) + 4) + 2) + 1) & 0xaa) >> 1) | ((((((((((0) + 64) + 32) + 16) + 8) + 4) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((((0) + 64) + 32) + 16) + 8) + 4) + 2) + 1) & 0xaa) >> 1) | ((((((((((0) + 64) + 32) + 16) + 8) + 4) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((0) + 127) & 0xaa) >> 1) | (((0) + 127) & 0x55) << 1)) & 0xcc) >> 2) | ((((((0) + 127) & 0xaa) >> 1) | (((0) + 127) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((0) + 127) & 0xaa) >> 1) | (((0) + 127) & 0x55) << 1)) & 0xcc) >> 2) | ((((((0) + 127) & 0xaa) >> 1) | (((0) + 127) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((0) + 127) + 1) & 0xaa) >> 1) | ((((0) + 127) + 1) & 0x55) << 1)) & 0xcc) >> 2) | (((((((0) + 127) + 1) & 0xaa) >> 1) | ((((0) + 127) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((0) + 127) + 1) & 0xaa) >> 1) | ((((0) + 127) + 1) & 0x55) << 1)) & 0xcc) >> 2) | (((((((0) + 127) + 1) & 0xaa) >> 1) | ((((0) + 127) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((0) + 127) + 2) & 0xaa) >> 1) | ((((0) + 127) + 2) & 0x55) << 1)) & 0xcc) >> 2) | (((((((0) + 127) + 2) & 0xaa) >> 1) | ((((0) + 127) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((0) + 127) + 2) & 0xaa) >> 1) | ((((0) + 127) + 2) & 0x55) << 1)) & 0xcc) >> 2) | (((((((0) + 127) + 2) & 0xaa) >> 1) | ((((0) + 127) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((0) + 127) + 2) + 1) & 0xaa) >> 1) | (((((0) + 127) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((0) + 127) + 2) + 1) & 0xaa) >> 1) | (((((0) + 127) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((((0) + 127) + 2) + 1) & 0xaa) >> 1) | (((((0) + 127) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((0) + 127) + 2) + 1) & 0xaa) >> 1) | (((((0) + 127) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((0) + 127) + 4) & 0xaa) >> 1) | ((((0) + 127) + 4) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((0) + 127) + 4) & 0xaa) >> 1) | ((((0) + 127) + 4) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((0) + 127) + 4) & 0xaa) >> 1) | ((((0) + 127) + 4) & 0x55) << 1)) & 0xcc) >> 2) | (((((((0) + 127) + 4) & 0xaa) >> 1) | ((((0) + 127) + 4) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((((0) + 127) + 4) + 1) & 0xaa) >> 1) | (((((0) + 127) + 4) + 1) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((0) + 127) + 4) + 1) & 0xaa) >> 1) | (((((0) + 127) + 4) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((0) + 127) + 4) + 1) & 0xaa) >> 1) | (((((0) + 127) + 4) + 1) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((0) + 127) + 4) + 1) & 0xaa) >> 1) | (((((0) + 127) + 4) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((((0) + 127) + 4) + 2) & 0xaa) >> 1) | (((((0) + 127) + 4) + 2) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((0) + 127) + 4) + 2) & 0xaa) >> 1) | (((((0) + 127) + 4) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((0) + 127) + 4) + 2) & 0xaa) >> 1) | (((((0) + 127) + 4) + 2) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((0) + 127) + 4) + 2) & 0xaa) >> 1) | (((((0) + 127) + 4) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((((0) + 127) + 4) + 2) + 1) & 0xaa) >> 1) | ((((((0) + 127) + 4) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((0) + 127) + 4) + 2) + 1) & 0xaa) >> 1) | ((((((0) + 127) + 4) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((((((0) + 127) + 4) + 2) + 1) & 0xaa) >> 1) | ((((((0) + 127) + 4) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((0) + 127) + 4) + 2) + 1) & 0xaa) >> 1) | ((((((0) + 127) + 4) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((((0) + 127) + 4) + 2) + 1) & 0xaa) >> 1) | ((((((0) + 127) + 4) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((0) + 127) + 4) + 2) + 1) & 0xaa) >> 1) | ((((((0) + 127) + 4) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((((((0) + 127) + 4) + 2) + 1) & 0xaa) >> 1) | ((((((0) + 127) + 4) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((0) + 127) + 4) + 2) + 1) & 0xaa) >> 1) | ((((((0) + 127) + 4) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((

(0) + 127) + 8) & 0xaa) >> 1) | (((((0) + 127) + 8) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((0) + 127) + 8) & 0xaa) >> 1) | (((((0) + 127) + 8) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((0) + 127) + 8) & 0xaa) >> 1) | (((((0) + 127) + 8) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((0) + 127) + 8) & 0xaa) >> 1) | (((((0) + 127) + 8) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((((0) + 127) + 8) + 1) & 0xaa) >> 1) | ((((((0) + 127) + 8) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((0) + 127) + 8) + 1) & 0xaa) >> 1) | ((((((0) + 127) + 8) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((0) + 127) + 8) + 1) & 0xaa) >> 1) | ((((((0) + 127) + 8) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((0) + 127) + 8) + 1) & 0xaa) >> 1) | ((((((0) + 127) + 8) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((((0) + 127) + 8) + 2) & 0xaa) >> 1) | (((((0) + 127) + 8) + 2) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((0) + 127) + 8) + 2) & 0xaa) >> 1) | (((((0) + 127) + 8) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((((0) + 127) + 8) + 2) & 0xaa) >> 1) | (((((0) + 127) + 8) + 2) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((0) + 127) + 8) + 2) & 0xaa) >> 1) | (((((0) + 127) + 8) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((((0) + 127) + 8) + 2) + 1) & 0xaa) >> 1) | (((((((0) + 127) + 8) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((0) + 127) + 8) + 2) + 1) & 0xaa) >> 1) | (((((((0) + 127) + 8) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((((((0) + 127) + 8) + 2) + 1) & 0xaa) >> 1) | (((((((0) + 127) + 8) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((0) + 127) + 8) + 2) + 1) & 0xaa) >> 1) | (((((((0) + 127) + 8) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((((0) + 127) + 8) + 4) & 0xaa) >> 1) | ((((((0) + 127) + 8) + 4) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((0) + 127) + 8) + 4) & 0xaa) >> 1) | ((((((0) + 127) + 8) + 4) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((0) + 127) + 8) + 4) & 0xaa) >> 1) | ((((((0) + 127) + 8) + 4) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((0) + 127) + 8) + 4) & 0xaa) >> 1) | ((((((0) + 127) + 8) + 4) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((((((0) + 127) + 8) + 4) + 1) & 0xaa) >> 1) | (((((((0) + 127) + 8) + 4) + 1) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((0) + 127) + 8) + 4) + 1) & 0xaa) >> 1) | (((((((0) + 127) + 8) + 4) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((((((0) + 127) + 8) + 4) + 1) & 0xaa) >> 1) | (((((((0) + 127) + 8) + 4) + 1) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((0) + 127) + 8) + 4) + 1) & 0xaa) >> 1) | (((((((0) + 127) + 8) + 4) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((((((0) + 127) + 8) + 4) + 2) & 0xaa) >> 1) | (((((((0) + 127) + 8) + 4) + 2) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((0) + 127) + 8) + 4) + 2) & 0xaa) >> 1) | (((((((0) + 127) + 8) + 4) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((((((0) + 127) + 8) + 4) + 2) & 0xaa) >> 1) | (((((((0) + 127) + 8) + 4) + 2) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((0) + 127) + 8) + 4) + 2) & 0xaa) >> 1) | (((((((0) + 127) + 8) + 4) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((((((0) + 127) + 8) + 4) + 2) + 1) & 0xaa) >> 1) | ((((((((0) + 127) + 8) + 4) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((0) + 127) + 8) + 4) + 2) + 1) & 0xaa) >> 1) | ((((((((0) + 127) + 8) + 4) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((((0) + 127) + 8) + 4) + 2) + 1) & 0xaa) >> 1) | ((((((((0) + 127) + 8) + 4) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((0) + 127) + 8) + 4) + 2) + 1) & 0xaa) >> 1) | ((((((((0) + 127) + 8) + 4) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((((0) + 127) + 16) & 0xaa) >> 1) | ((((0) + 127) + 16) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((0) + 127) + 16) & 0xaa) >> 1) | ((((0) + 127) + 16) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((0) + 127) + 16) & 0xaa) >> 1) | ((((0) + 127) + 16) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((0) + 127) + 16) & 0xaa) >> 1) | ((((0) + 127) + 16) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((((0) + 127) + 16) + 1) & 0xaa) >> 1) | (((((0) + 127) + 16) + 1) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((0) + 127) + 16) + 1) & 0xaa) >> 1) | (((((0) + 127) + 16) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((((0) + 127) + 16) + 1) & 0xaa) >> 1) | (((((0) + 127) + 16) + 1) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((0) + 127) + 16) + 1) & 0xaa) >> 1) | (((((0) + 127) + 16) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((((0) + 127) + 16) + 2) & 0xaa) >> 1) | (((((0) + 127) + 16) + 2) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((0) + 127) + 16) + 2) & 0xaa) >> 1) | (((((0) + 127) + 16) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((((0) + 127) + 16) + 2) & 0xaa) >> 1) | (((((0) + 127) + 16) + 2) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((0) + 127) + 16) + 2) & 0xaa) >> 1) | (((((0) + 127) + 16) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((((0) + 127) + 16) + 2) + 1) & 0xaa) >> 1) | (((((((0) + 127) + 16) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((0) + 127) + 16) + 2) + 1) & 0xaa) >> 1) | (((((((0) + 127) + 16) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((((((0) + 127) + 16) + 2) + 1) & 0xaa) >> 1) | (((((((0) + 127) + 16) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((0) + 127) + 16) + 2) + 1) & 0xaa) >> 1) | (((((((0) + 127) + 16) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((((0) + 127) + 16) + 4) & 0xaa) >> 1) | ((((((0) + 127) + 16) + 4) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((0) + 127) + 16) + 4) & 0xaa) >> 1) | ((((((0) + 127) + 16) + 4) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((0) + 127) + 16) + 4) & 0xaa) >> 1) | ((((((0) + 127) + 16) + 4) & 0x55) << 1)) & 0xcc) >> 2) | ((((((

(((0) + 127) + 16) + 4) & 0xaa) >> 1) | ((((((0) + 127) + 16) + 4) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((((0) + 127) + 16) + 4) + 1) & 0xaa) >> 1) | (((((((0) + 127) + 16) + 4) + 1) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((0) + 127) + 16) + 4) + 1) & 0xaa) >> 1) | (((((((0) + 127) + 16) + 4) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((((((0) + 127) + 16) + 4) + 1) & 0xaa) >> 1) | (((((((0) + 127) + 16) + 4) + 1) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((0) + 127) + 16) + 4) + 1) & 0xaa) >> 1) | (((((((0) + 127) + 16) + 4) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((((0) + 127) + 16) + 4) + 2) & 0xaa) >> 1) | (((((((0) + 127) + 16) + 4) + 2) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((0) + 127) + 16) + 4) + 2) & 0xaa) >> 1) | (((((((0) + 127) + 16) + 4) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((((((0) + 127) + 16) + 4) + 2) & 0xaa) >> 1) | (((((((0) + 127) + 16) + 4) + 2) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((0) + 127) + 16) + 4) + 2) & 0xaa) >> 1) | (((((((0) + 127) + 16) + 4) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((((((0) + 127) + 16) + 4) + 2) + 1) & 0xaa) >> 1) | ((((((((0) + 127) + 16) + 4) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((0) + 127) + 16) + 4) + 2) + 1) & 0xaa) >> 1) | (((((((((0) + 127) + 16) + 4) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((((0) + 127) + 16) + 4) + 2) + 1) & 0xaa) >> 1) | ((((((((0) + 127) + 16) + 4) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((0) + 127) + 16) + 4) + 2) + 1) & 0xaa) >> 1) | ((((((((0) + 127) + 16) + 4) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((((0) + 127) + 16) + 8) & 0xaa) >> 1) | ((((((0) + 127) + 16) + 8) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((0) + 127) + 16) + 8) & 0xaa) >> 1) | ((((((0) + 127) + 16) + 8) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((0) + 127) + 16) + 8) & 0xaa) >> 1) | ((((((0) + 127) + 16) + 8) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((0) + 127) + 16) + 8) & 0xaa) >> 1) | ((((((0) + 127) + 16) + 8) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((((0) + 127) + 16) + 8) + 1) & 0xaa) >> 1) | (((((((0) + 127) + 16) + 8) + 1) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((0) + 127) + 16) + 8) + 1) & 0xaa) >> 1) | (((((((0) + 127) + 16) + 8) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((((((0) + 127) + 16) + 8) + 1) & 0xaa) >> 1) | (((((((0) + 127) + 16) + 8) + 1) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((0) + 127) + 16) + 8) + 1) & 0xaa) >> 1) | (((((((0) + 127) + 16) + 8) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((((0) + 127) + 16) + 8) + 2) & 0xaa) >> 1) | (((((((0) + 127) + 16) + 8) + 2) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((0) + 127) + 16) + 8) + 2) & 0xaa) >> 1) | (((((((0) + 127) + 16) + 8) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((((((0) + 127) + 16) + 8) + 2) & 0xaa) >> 1) | (((((((0) + 127) + 16) + 8) + 2) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((0) + 127) + 16) + 8) + 2) & 0xaa) >> 1) | (((((((0) + 127) + 16) + 8) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((((((0) + 127) + 16) + 8) + 2) + 1) & 0xaa) >> 1) | ((((((((0) + 127) + 16) + 8) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((0) + 127) + 16) + 8) + 2) + 1) & 0xaa) >> 1) | ((((((((0) + 127) + 16) + 8) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((((0) + 127) + 16) + 8) + 2) + 1) & 0xaa) >> 1) | ((((((((0) + 127) + 16) + 8) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((0) + 127) + 16) + 8) + 2) + 1) & 0xaa) >> 1) | ((((((((0) + 127) + 16) + 8) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((((0) + 127) + 16) + 8) + 4) & 0xaa) >> 1) | (((((((0) + 127) + 16) + 8) + 4) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((0) + 127) + 16) + 8) + 4) & 0xaa) >> 1) | (((((((0) + 127) + 16) + 8) + 4) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((((((0) + 127) + 16) + 8) + 4) & 0xaa) >> 1) | (((((((0) + 127) + 16) + 8) + 4) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((0) + 127) + 16) + 8) + 4) & 0xaa) >> 1) | (((((((0) + 127) + 16) + 8) + 4) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((((((0) + 127) + 16) + 8) + 4) + 1) & 0xaa) >> 1) | ((((((((0) + 127) + 16) + 8) + 4) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((0) + 127) + 16) + 8) + 4) + 1) & 0xaa) >> 1) | ((((((((0) + 127) + 16) + 8) + 4) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((((0) + 127) + 16) + 8) + 4) + 1) & 0xaa) >> 1) | ((((((((0) + 127) + 16) + 8) + 4) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((0) + 127) + 16) + 8) + 4) + 1) & 0xaa) >> 1) | (((((((((0) + 127) + 16) + 8) + 4) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((((((0) + 127) + 16) + 8) + 4) + 2) & 0xaa) >> 1) | ((((((((0) + 127) + 16) + 8) + 4) + 2) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((0) + 127) + 16) + 8) + 4) + 2) & 0xaa) >> 1) | ((((((((0) + 127) + 16) + 8) + 4) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((((0) + 127) + 16) + 8) + 4) + 2) & 0xaa) >> 1) | ((((((((0) + 127) + 16) + 8) + 4) + 2) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((0) + 127) + 16) + 8) + 4) + 2) & 0xaa) >> 1) | ((((((((0) + 127) + 16) + 8) + 4) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((((((0) + 127) + 16) + 8) + 4) + 2) + 1) & 0xaa) >> 1) | (((((((((0) + 127) + 16) + 8) + 4) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((((0) + 127) + 16) + 8) + 4) + 2) + 1) & 0xaa) >> 1) | (((((((((0) + 127) + 16) + 8) + 4) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((((((((0) + 127) + 16) + 8) + 4) + 2) + 1) & 0xaa) >> 1) | (((((((((0) + 127) + 16) + 8) + 4) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((((0) + 127) + 16) + 8) + 4) + 2) + 1) & 0xaa) >> 1) | (((((((((0) + 127) + 16) + 8) + 4) + 2) + 1) & 0x55) << 1)) & 0x33)

<< 2)) & 0x0f) << 4)), (((((((((((((0) + 127) + 32) & 0xaa) >> 1) | (((((0) + 127) + 32) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((0) + 127) + 32) & 0xaa) >> 1) | (((((0) + 127) + 32) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((((0) + 127) + 32) & 0xaa) >> 1) | (((((0) + 127) + 32) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((0) + 127) + 32) & 0xaa) >> 1) | (((((0) + 127) + 32) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((((0) + 127) + 32) + 1) & 0xaa) >> 1) | ((((((0) + 127) + 32) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((0) + 127) + 32) + 1) & 0xaa) >> 1) | ((((((0) + 127) + 32) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((0) + 127) + 32) + 1) & 0xaa) >> 1) | ((((((0) + 127) + 32) + 1) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((0) + 127) + 32) + 1) & 0xaa) >> 1) | ((((((0) + 127) + 32) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((((0) + 127) + 32) + 2) & 0xaa) >> 1) | ((((((0) + 127) + 32) + 2) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((0) + 127) + 32) + 2) & 0xaa) >> 1) | ((((((0) + 127) + 32) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((0) + 127) + 32) + 2) & 0xaa) >> 1) | ((((((0) + 127) + 32) + 2) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((0) + 127) + 32) + 2) & 0xaa) >> 1) | ((((((0) + 127) + 32) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((((((0) + 127) + 32) + 2) + 1) & 0xaa) >> 1) | (((((((0) + 127) + 32) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((0) + 127) + 32) + 2) + 1) & 0xaa) >> 1) | (((((((0) + 127) + 32) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((((0) + 127) + 32) + 2) + 1) & 0xaa) >> 1) | (((((((0) + 127) + 32) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((0) + 127) + 32) + 2) + 1) & 0xaa) >> 1) | (((((((0) + 127) + 32) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((((0) + 127) + 32) + 4) & 0xaa) >> 1) | ((((((0) + 127) + 32) + 4) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((0) + 127) + 32) + 4) & 0xaa) >> 1) | ((((((0) + 127) + 32) + 4) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((0) + 127) + 32) + 4) & 0xaa) >> 1) | ((((((0) + 127) + 32) + 4) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((0) + 127) + 32) + 4) & 0xaa) >> 1) | ((((((0) + 127) + 32) + 4) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((((((0) + 127) + 32) + 4) + 1) & 0xaa) >> 1) | (((((((0) + 127) + 32) + 4) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((0) + 127) + 32) + 4) + 1) & 0xaa) >> 1) | (((((((0) + 127) + 32) + 4) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((((((0) + 127) + 32) + 4) + 1) & 0xaa) >> 1) | (((((((0) + 127) + 32) + 4) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((0) + 127) + 32) + 4) + 1) & 0xaa) >> 1) | (((((((0) + 127) + 32) + 4) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((((((0) + 127) + 32) + 4) + 2) & 0xaa) >> 1) | (((((((0) + 127) + 32) + 4) + 2) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((0) + 127) + 32) + 4) + 2) & 0xaa) >> 1) | (((((((0) + 127) + 32) + 4) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((((0) + 127) + 32) + 4) + 2) & 0xaa) >> 1) | (((((((0) + 127) + 32) + 4) + 2) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((0) + 127) + 32) + 4) + 2) & 0xaa) >> 1) | (((((((0) + 127) + 32) + 4) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((((((((0) + 127) + 32) + 4) + 2) + 1) & 0xaa) >> 1) | ((((((((0) + 127) + 32) + 4) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((((0) + 127) + 32) + 4) + 2) + 1) & 0xaa) >> 1) | ((((((((0) + 127) + 32) + 4) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((((((((0) + 127) + 32) + 4) + 2) + 1) & 0xaa) >> 1) | ((((((((0) + 127) + 32) + 4) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((((0) + 127) + 32) + 4) + 2) + 1) & 0xaa) >> 1) | ((((((((0) + 127) + 32) + 4) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((((((0) + 127) + 32) + 8) & 0xaa) >> 1) | ((((((0) + 127) + 32) + 8) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((0) + 127) + 32) + 8) & 0xaa) >> 1) | ((((((0) + 127) + 32) + 8) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((0) + 127) + 32) + 8) & 0xaa) >> 1) | ((((((0) + 127) + 32) + 8) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((0) + 127) + 32) + 8) & 0xaa) >> 1) | ((((((0) + 127) + 32) + 8) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((((((0) + 127) + 32) + 8) + 1) & 0xaa) >> 1) | (((((((0) + 127) + 32) + 8) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((0) + 127) + 32) + 8) + 1) & 0xaa) >> 1) | (((((((0) + 127) + 32) + 8) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((((((0) + 127) + 32) + 8) + 1) & 0xaa) >> 1) | (((((((0) + 127) + 32) + 8) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((0) + 127) + 32) + 8) + 1) & 0xaa) >> 1) | (((((((0) + 127) + 32) + 8) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((((((0) + 127) + 32) + 8) + 2) & 0xaa) >> 1) | (((((((0) + 127) + 32) + 8) + 2) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((0) + 127) + 32) + 8) + 2) & 0xaa) >> 1) | (((((((0) + 127) + 32) + 8) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((((((0) + 127) + 32) + 8) + 2) & 0xaa) >> 1) | (((((((0) + 127) + 32) + 8) + 2) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((0) + 127) + 32) + 8) + 2) & 0xaa) >> 1) | (((((((0) + 127) + 32) + 8) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((((((0) + 127) + 32) + 8) + 2) + 1) & 0xaa) >> 1) | ((((((((0) + 127) + 32) + 8) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((((0) + 127) + 32) + 8) + 2) + 1) & 0xaa) >> 1) | ((((((((0) + 127) + 32) + 8) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((((((((0) + 127) + 32) + 8) + 2) + 1) & 0xaa) >> 1) | ((((((((0) + 127) + 32) + 8) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((((0) + 127) + 32) + 8) + 2) + 1) & 0xaa) >> 1) | ((((((((0) + 127) + 32) + 8) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((((((0) + 127) + 32) + 8) + 4) & 0xaa)

>> 1) | (((((((0) + 127) + 32) + 8) + 4) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((0) + 127) + 32) + 8) + 4) & 0xaa) >> 1) | (((((((0) + 127) + 32) + 8) + 4) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((((((0) + 127) + 32) + 8) + 4) & 0xaa) >> 1) | (((((((0) + 127) + 32) + 8) + 4) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((0) + 127) + 32) + 8) + 4) & 0xaa) >> 1) | (((((((0) + 127) + 32) + 8) + 4) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((((((0) + 127) + 32) + 8) + 4) + 1) & 0xaa) >> 1) | ((((((((0) + 127) + 32) + 8) + 4) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((0) + 127) + 32) + 8) + 4) + 1) & 0xaa) >> 1) | ((((((((0) + 127) + 32) + 8) + 4) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((((0) + 127) + 32) + 8) + 4) + 1) & 0xaa) >> 1) | ((((((((0) + 127) + 32) + 8) + 4) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((0) + 127) + 32) + 8) + 4) + 1) & 0xaa) >> 1) | ((((((((0) + 127) + 32) + 8) + 4) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((((((0) + 127) + 32) + 8) + 4) + 2) & 0xaa) >> 1) | ((((((((0) + 127) + 32) + 8) + 4) + 2) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((0) + 127) + 32) + 8) + 4) + 2) & 0xaa) >> 1) | ((((((((0) + 127) + 32) + 8) + 4) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((((0) + 127) + 32) + 8) + 4) + 2) & 0xaa) >> 1) | ((((((((0) + 127) + 32) + 8) + 4) + 2) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((0) + 127) + 32) + 8) + 4) + 2) & 0xaa) >> 1) | ((((((((0) + 127) + 32) + 8) + 4) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((((((0) + 127) + 32) + 8) + 4) + 2) + 1) & 0xaa) >> 1) | (((((((((0) + 127) + 32) + 8) + 4) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((((0) + 127) + 32) + 8) + 4) + 2) + 1) & 0xaa) >> 1) | (((((((((0) + 127) + 32) + 8) + 4) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((((((((0) + 127) + 32) + 8) + 4) + 2) + 1) & 0xaa) >> 1) | (((((((((0) + 127) + 32) + 8) + 4) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((((0) + 127) + 32) + 8) + 4) + 2) + 1) & 0xaa) >> 1) | (((((((((0) + 127) + 32) + 8) + 4) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((((0) + 127) + 32) + 16) & 0xaa) >> 1) | ((((((0) + 127) + 32) + 16) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((0) + 127) + 32) + 16) & 0xaa) >> 1) | ((((((0) + 127) + 32) + 16) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((0) + 127) + 32) + 16) & 0xaa) >> 1) | ((((((0) + 127) + 32) + 16) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((0) + 127) + 32) + 16) & 0xaa) >> 1) | ((((((0) + 127) + 32) + 16) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((((0) + 127) + 32) + 16) + 1) & 0xaa) >> 1) | ((((((0) + 127) + 32) + 16) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((0) + 127) + 32) + 16) + 1) & 0xaa) >> 1) | ((((((0) + 127) + 32) + 16) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((0) + 127) + 32) + 16) + 1) & 0xaa) >> 1) | ((((((0) + 127) + 32) + 16) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((0) + 127) + 32) + 16) + 1) & 0xaa) >> 1) | ((((((0) + 127) + 32) + 16) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((((((0) + 127) + 32) + 16) + 2) & 0xaa) >> 1) | (((((((0) + 127) + 32) + 16) + 2) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((0) + 127) + 32) + 16) + 2) & 0xaa) >> 1) | (((((((0) + 127) + 32) + 16) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((((((0) + 127) + 32) + 16) + 2) & 0xaa) >> 1) | (((((((0) + 127) + 32) + 16) + 2) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((0) + 127) + 32) + 16) + 2) & 0xaa) >> 1) | (((((((0) + 127) + 32) + 16) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((((((0) + 127) + 32) + 16) + 2) + 1) & 0xaa) >> 1) | ((((((((0) + 127) + 32) + 16) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((0) + 127) + 32) + 16) + 2) + 1) & 0xaa) >> 1) | (((((((((0) + 127) + 32) + 16) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((((0) + 127) + 32) + 16) + 2) + 1) & 0xaa) >> 1) | ((((((((0) + 127) + 32) + 16) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((0) + 127) + 32) + 16) + 2) + 1) & 0xaa) >> 1) | ((((((((0) + 127) + 32) + 16) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((((((0) + 127) + 32) + 16) + 4) & 0xaa) >> 1) | (((((((0) + 127) + 32) + 16) + 4) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((0) + 127) + 32) + 16) + 4) & 0xaa) >> 1) | (((((((0) + 127) + 32) + 16) + 4) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((((((0) + 127) + 32) + 16) + 4) & 0xaa) >> 1) | (((((((0) + 127) + 32) + 16) + 4) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((0) + 127) + 32) + 16) + 4) & 0xaa) >> 1) | (((((((0) + 127) + 32) + 16) + 4) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((((((0) + 127) + 32) + 16) + 4) + 1) & 0xaa) >> 1) | ((((((((0) + 127) + 32) + 16) + 4) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((0) + 127) + 32) + 16) + 4) + 1) & 0xaa) >> 1) | ((((((((0) + 127) + 32) + 16) + 4) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((((0) + 127) + 32) + 16) + 4) + 1) & 0xaa) >> 1) | ((((((((0) + 127) + 32) + 16) + 4) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((0) + 127) + 32) + 16) + 4) + 1) & 0xaa) >> 1) | ((((((((0) + 127) + 32) + 16) + 4) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((((((0) + 127) + 32) + 16) + 4) + 2) & 0xaa) >> 1) | ((((((((0) + 127) + 32) + 16) + 4) + 2) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((0) + 127) + 32) + 16) + 4) + 2) & 0xaa) >> 1) | ((((((((0) + 127) + 32) + 16) + 4) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((((((0) + 127) + 32) + 16) + 4) + 2) & 0xaa) >> 1) | ((((((((0) + 127) + 32) + 16) + 4) + 2) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((0) + 127) + 32) + 16) + 4) + 2) & 0xaa) >> 1) | ((((((((0) + 127) + 32) + 16) + 4) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((((((((0) + 127) + 32) +

16) + 4) + 2) + 1) & 0xaa) >> 1) | (((((((((0) + 127) + 32) + 16) + 4) + 2) + 1) & 0x55) << 1)) &
0xcc) >> 2) | ((((((((((((0) + 127) + 32) + 16) + 4) + 2) + 1) & 0xaa) >> 1) | (((((((((0) + 127) +
32) + 16) + 4) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((((((((0) + 127) + 32)
+ 16) + 4) + 2) + 1) & 0xaa) >> 1) | (((((((((0) + 127) + 32) + 16) + 4) + 2) + 1) & 0x55) << 1)) &
0xcc) >> 2) | ((((((((((((0) + 127) + 32) + 16) + 4) + 2) + 1) & 0xaa) >> 1) | (((((((((0) + 127) +
32) + 16) + 4) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((((0) + 127) + 32) +
16) + 8) & 0xaa) >> 1) | (((((((0) + 127) + 32) + 16) + 8) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((0
) + 127) + 32) + 16) + 8) & 0xaa) >> 1) | (((((((0) + 127) + 32) + 16) + 8) & 0x55) << 1)) & 0x33)
<< 2)) & 0xf0) >> 4) | (((((((((((((0) + 127) + 32) + 16) + 8) & 0xaa) >> 1) | (((((((0) + 127) + 32)
+ 16) + 8) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((0) + 127) + 32) + 16) + 8) & 0xaa) >> 1) | ((((((
(0) + 127) + 32) + 16) + 8) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((((0) + 127) +
32) + 16) + 8) + 1) & 0xaa) >> 1) | ((((((((0) + 127) + 32) + 16) + 8) + 1) & 0x55) << 1)) & 0xcc)
>> 2) | (((((((((((0) + 127) + 32) + 16) + 8) + 1) & 0xaa) >> 1) | ((((((((0) + 127) + 32) + 16) + 8
) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((((0) + 127) + 32) + 16) + 8) + 1) &
0xaa) >> 1) | ((((((((0) + 127) + 32) + 16) + 8) + 1) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((0) +
127) + 32) + 16) + 8) + 1) & 0xaa) >> 1) | ((((((((0) + 127) + 32) + 16) + 8) + 1) & 0x55) << 1)) &
0x33) << 2)) & 0x0f) << 4)), ((((((((((((((0) + 127) + 32) + 16) + 8) + 2) & 0xaa) >> 1) | ((((((((0
) + 127) + 32) + 16) + 8) + 2) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((0) + 127) + 32) + 16) + 8) +
2) & 0xaa) >> 1) | ((((((((0) + 127) + 32) + 16) + 8) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >>
4) | ((((((((((((((0) + 127) + 32) + 16) + 8) + 2) & 0xaa) >> 1) | ((((((((0) + 127) + 32) + 16) + 8)
+ 2) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((0) + 127) + 32) + 16) + 8) + 2) & 0xaa) >> 1) | (((((
(((0) + 127) + 32) + 16) + 8) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((((((0)
+ 127) + 32) + 16) + 8) + 2) + 1) & 0xaa) >> 1) | (((((((((0) + 127) + 32) + 16) + 8) + 2) + 1) &
0x55) << 1)) & 0xcc) >> 2) | ((((((((((((0) + 127) + 32) + 16) + 8) + 2) + 1) & 0xaa) >> 1) | (((((((
((0) + 127) + 32) + 16) + 8) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((((((((0
) + 127) + 32) + 16) + 8) + 2) + 1) & 0xaa) >> 1) | (((((((((0) + 127) + 32) + 16) + 8) + 2) + 1) &
0x55) << 1)) & 0xcc) >> 2) | ((((((((((((0) + 127) + 32) + 16) + 8) + 2) + 1) & 0xaa) >> 1) | (((((((
((0) + 127) + 32) + 16) + 8) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((((0)
+ 127) + 32) + 16) + 8) + 4) & 0xaa) >> 1) | ((((((((0) + 127) + 32) + 16) + 8) + 4) & 0x55) << 1))
& 0xcc) >> 2) | (((((((((((0) + 127) + 32) + 16) + 8) + 4) & 0xaa) >> 1) | ((((((((0) + 127) + 32) +
16) + 8) + 4) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((((0) + 127) + 32) + 16) + 8) +
4) & 0xaa) >> 1) | ((((((((0) + 127) + 32) + 16) + 8) + 4) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((
0) + 127) + 32) + 16) + 8) + 4) & 0xaa) >> 1) | ((((((((0) + 127) + 32) + 16) + 8) + 4) & 0x55) <<
1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((((((0) + 127) + 32) + 16) + 8) + 4) + 1) & 0xaa) >> 1) |
(((((((((0) + 127) + 32) + 16) + 8) + 4) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((0) + 127)
+ 32) + 16) + 8) + 4) + 1) & 0xaa) >> 1) | (((((((((0) + 127) + 32) + 16) + 8) + 4) + 1) & 0x55) <<
1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((((((((0) + 127) + 32) + 16) + 8) + 4) + 1) & 0xaa) >> 1) |
(((((((((0) + 127) + 32) + 16) + 8) + 4) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((0) + 127)
+ 32) + 16) + 8) + 4) + 1) & 0xaa) >> 1) | (((((((((0) + 127) + 32) + 16) + 8) + 4) + 1) & 0x55) <<
1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((((((0) + 127) + 32) + 16) + 8) + 4) + 2) & 0xaa) >> 1) |
(((((((((0) + 127) + 32) + 16) + 8) + 4) + 2) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((0) + 127)
+ 32) + 16) + 8) + 4) + 2) & 0xaa) >> 1) | (((((((((0) + 127) + 32) + 16) + 8) + 4) + 2) & 0x55) <<
1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((((((((0) + 127) + 32) + 16) + 8) + 4) + 2) & 0xaa) >> 1) |
(((((((((0) + 127) + 32) + 16) + 8) + 4) + 2) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((0) + 127)
+ 32) + 16) + 8) + 4) + 2) & 0xaa) >> 1) | (((((((((0) + 127) + 32) + 16) + 8) + 4) + 2) & 0x55) <<
1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((((((0) + 127) + 32) + 16) + 8) + 4) + 2) + 1) & 0xaa)
>> 1) | ((((((((((0) + 127) + 32) + 16) + 8) + 4) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((
(((0) + 127) + 32) + 16) + 8) + 4) + 2) + 1) & 0xaa) >> 1) | ((((((((((0) + 127) + 32) + 16) + 8) +
4) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((((((0) + 127) + 32) + 16) + 8)
+ 4) + 2) + 1) & 0xaa) >> 1) | ((((((((((0) + 127) + 32) + 16) + 8) + 4) + 2) + 1) & 0x55) << 1)) &
0xcc) >> 2) | (((((((((((((0) + 127) + 32) + 16) + 8) + 4) + 2) + 1) & 0xaa) >> 1) | ((((((((((0) +
127) + 32) + 16) + 8) + 4) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((0) + 127
) + 64) & 0xaa) >> 1) | (((((0) + 127) + 64) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((0) + 127) + 64) &
0xaa) >> 1) | (((((0) + 127) + 64) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((((0) + 127
) + 64) & 0xaa) >> 1) | (((((0) + 127) + 64) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((0) + 127) + 64)
& 0xaa) >> 1) | (((((0) + 127) + 64) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((0) +
127) + 64) + 1) & 0xaa) >> 1) | ((((((0) + 127) + 64) + 1) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((0)
+ 127) + 64) + 1) & 0xaa) >> 1) | ((((((0) + 127) + 64) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0)

>> 4) | ((((((((((((0) + 127) + 64) + 1) & 0xaa) >> 1) | ((((((0) + 127) + 64) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((0) + 127) + 64) + 1) & 0xaa) >> 1) | ((((((0) + 127) + 64) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((((0) + 127) + 64) + 2) & 0xaa) >> 1) | ((((((0) + 127) + 64) + 2) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((0) + 127) + 64) + 2) & 0xaa) >> 1) | ((((((0) + 127) + 64) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((0) + 127) + 64) + 2) & 0xaa) >> 1) | ((((((0) + 127) + 64) + 2) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((0) + 127) + 64) + 2) & 0xaa) >> 1) | ((((((0) + 127) + 64) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((((((0) + 127) + 64) + 2) + 1) & 0xaa) >> 1) | (((((((0) + 127) + 64) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((0) + 127) + 64) + 2) + 1) & 0xaa) >> 1) | (((((((0) + 127) + 64) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((((0) + 127) + 64) + 2) + 1) & 0xaa) >> 1) | (((((((0) + 127) + 64) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((0) + 127) + 64) + 2) + 1) & 0xaa) >> 1) | (((((((0) + 127) + 64) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((((0) + 127) + 64) + 4) & 0xaa) >> 1) | ((((((0) + 127) + 64) + 4) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((0) + 127) + 64) + 4) & 0xaa) >> 1) | ((((((0) + 127) + 64) + 4) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((0) + 127) + 64) + 4) & 0xaa) >> 1) | ((((((0) + 127) + 64) + 4) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((0) + 127) + 64) + 4) & 0xaa) >> 1) | ((((((0) + 127) + 64) + 4) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((((((0) + 127) + 64) + 4) + 1) & 0xaa) >> 1) | (((((((0) + 127) + 64) + 4) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((0) + 127) + 64) + 4) + 1) & 0xaa) >> 1) | (((((((0) + 127) + 64) + 4) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((((0) + 127) + 64) + 4) + 1) & 0xaa) >> 1) | (((((((0) + 127) + 64) + 4) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((0) + 127) + 64) + 4) + 1) & 0xaa) >> 1) | (((((((0) + 127) + 64) + 4) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((((((0) + 127) + 64) + 4) + 2) & 0xaa) >> 1) | (((((((0) + 127) + 64) + 4) + 2) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((0) + 127) + 64) + 4) + 2) & 0xaa) >> 1) | (((((((0) + 127) + 64) + 4) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((((0) + 127) + 64) + 4) + 2) & 0xaa) >> 1) | (((((((0) + 127) + 64) + 4) + 2) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((0) + 127) + 64) + 4) + 2) & 0xaa) >> 1) | (((((((0) + 127) + 64) + 4) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((((((0) + 127) + 64) + 4) + 2) + 1) & 0xaa) >> 1) | ((((((((0) + 127) + 64) + 4) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((((0) + 127) + 64) + 4) + 2) + 1) & 0xaa) >> 1) | ((((((((0) + 127) + 64) + 4) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((((0) + 127) + 64) + 4) + 2) + 1) & 0xaa) >> 1) | (((((((0) + 127) + 64) + 4) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((0) + 127) + 64) + 4) + 2) + 1) & 0xaa) >> 1) | (((((((0) + 127) + 64) + 4) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((((0) + 127) + 64) + 8) & 0xaa) >> 1) | ((((((0) + 127) + 64) + 8) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((0) + 127) + 64) + 8) & 0xaa) >> 1) | ((((((0) + 127) + 64) + 8) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((0) + 127) + 64) + 8) & 0xaa) >> 1) | ((((((0) + 127) + 64) + 8) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((0) + 127) + 64) + 8) & 0xaa) >> 1) | ((((((0) + 127) + 64) + 8) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((((((0) + 127) + 64) + 8) + 1) & 0xaa) >> 1) | (((((((0) + 127) + 64) + 8) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((0) + 127) + 64) + 8) + 1) & 0xaa) >> 1) | (((((((0) + 127) + 64) + 8) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((((((0) + 127) + 64) + 8) + 1) & 0xaa) >> 1) | (((((((0) + 127) + 64) + 8) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((0) + 127) + 64) + 8) + 1) & 0xaa) >> 1) | (((((((0) + 127) + 64) + 8) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((((((0) + 127) + 64) + 8) + 2) & 0xaa) >> 1) | (((((((0) + 127) + 64) + 8) + 2) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((0) + 127) + 64) + 8) + 2) & 0xaa) >> 1) | (((((((0) + 127) + 64) + 8) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((((0) + 127) + 64) + 8) + 2) & 0xaa) >> 1) | (((((((0) + 127) + 64) + 8) + 2) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((0) + 127) + 64) + 8) + 2) & 0xaa) >> 1) | (((((((0) + 127) + 64) + 8) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((((((((0) + 127) + 64) + 8) + 2) + 1) & 0xaa) >> 1) | ((((((((0) + 127) + 64) + 8) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((((0) + 127) + 64) + 8) + 2) + 1) & 0xaa) >> 1) | ((((((((0) + 127) + 64) + 8) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((((0) + 127) + 64) + 8) + 2) + 1) & 0xaa) >> 1) | (((((((0) + 127) + 64) + 8) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((0) + 127) + 64) + 8) + 2) + 1) & 0xaa) >> 1) | (((((((0) + 127) + 64) + 8) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((((((0) + 127) + 64) + 8) + 4) & 0xaa) >> 1) | (((((((0) + 127) + 64) + 8) + 4) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((0) + 127) + 64) + 8) + 4) & 0xaa) >> 1) | (((((((0) + 127) + 64) + 8) + 4) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((((0) + 127) + 64) + 8) + 4) & 0xaa) >> 1) | (((((((0) + 127) + 64) + 8) + 4) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((0) + 127) + 64) + 8) + 4) & 0xaa) >> 1) | (((((((0) + 127) + 64) + 8) + 4) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((((((0) + 127) + 64) + 8) + 4) + 1) & 0xaa) >> 1) | ((((((((0) + 127) + 64) + 8) + 4) + 1) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((((0) + 127) + 64) + 8) +

4) + 1) & 0xaa) >> 1) | (((((((((0) + 127) + 64) + 8) + 4) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((((0) + 127) + 64) + 8) + 4) + 1) & 0xaa) >> 1) | ((((((((0) + 127) + 64) + 8) + 4) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((0) + 127) + 64) + 8) + 4) + 1) & 0xaa) >> 1) | (((((((((0) + 127) + 64) + 8) + 4) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((((((0) + 127) + 64) + 8) + 4) + 2) & 0xaa) >> 1) | ((((((((0) + 127) + 64) + 8) + 4) + 2) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((0) + 127) + 64) + 8) + 4) + 2) & 0xaa) >> 1) | ((((((((0) + 127) + 64) + 8) + 4) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((((0) + 127) + 64) + 8) + 4) + 2) & 0xaa) >> 1) | ((((((((0) + 127) + 64) + 8) + 4) + 2) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((0) + 127) + 64) + 8) + 4) + 2) & 0xaa) >> 1) | (((((((((0) + 127) + 64) + 8) + 4) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((((((0) + 127) + 64) + 8) + 4) + 2) + 1) & 0xaa) >> 1) | (((((((((0) + 127) + 64) + 8) + 4) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((0) + 127) + 64) + 8) + 4) + 2) + 1) & 0xaa) >> 1) | (((((((((0) + 127) + 64) + 8) + 4) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((((0) + 127) + 64) + 8) + 4) + 2) + 1) & 0xaa) >> 1) | (((((((((0) + 127) + 64) + 8) + 4) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((0) + 127) + 64) + 8) + 4) + 2) + 1) & 0xaa) >> 1) | (((((((((0) + 127) + 64) + 8) + 4) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((((0) + 127) + 64) + 16) & 0xaa) >> 1) | ((((((0) + 127) + 64) + 16) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((0) + 127) + 64) + 16) & 0xaa) >> 1) | ((((((0) + 127) + 64) + 16) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((((((0) + 127) + 64) + 16) & 0xaa) >> 1) | ((((((0) + 127) + 64) + 16) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((0) + 127) + 64) + 16) & 0xaa) >> 1) | ((((((0) + 127) + 64) + 16) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((((0) + 127) + 64) + 16) + 1) & 0xaa) >> 1) | (((((((0) + 127) + 64) + 16) + 1) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((0) + 127) + 64) + 16) + 1) & 0xaa) >> 1) | (((((((0) + 127) + 64) + 16) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((((0) + 127) + 64) + 16) + 1) & 0xaa) >> 1) | (((((((0) + 127) + 64) + 16) + 1) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((0) + 127) + 64) + 16) + 1) & 0xaa) >> 1) | (((((((0) + 127) + 64) + 16) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((((0) + 127) + 64) + 16) + 2) & 0xaa) >> 1) | (((((((0) + 127) + 64) + 16) + 2) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((0) + 127) + 64) + 16) + 2) & 0xaa) >> 1) | (((((((0) + 127) + 64) + 16) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((((0) + 127) + 64) + 16) + 2) & 0xaa) >> 1) | (((((((0) + 127) + 64) + 16) + 2) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((0) + 127) + 64) + 16) + 2) & 0xaa) >> 1) | (((((((0) + 127) + 64) + 16) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((((((0) + 127) + 64) + 16) + 2) + 1) & 0xaa) >> 1) | ((((((((0) + 127) + 64) + 16) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((0) + 127) + 64) + 16) + 2) + 1) & 0xaa) >> 1) | ((((((((0) + 127) + 64) + 16) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((((((0) + 127) + 64) + 16) + 2) + 1) & 0xaa) >> 1) | ((((((((0) + 127) + 64) + 16) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((0) + 127) + 64) + 16) + 2) + 1) & 0xaa) >> 1) | ((((((((0) + 127) + 64) + 16) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((((0) + 127) + 64) + 16) + 4) & 0xaa) >> 1) | (((((((0) + 127) + 64) + 16) + 4) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((0) + 127) + 64) + 16) + 4) & 0xaa) >> 1) | (((((((0) + 127) + 64) + 16) + 4) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((((0) + 127) + 64) + 16) + 4) & 0xaa) >> 1) | (((((((0) + 127) + 64) + 16) + 4) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((0) + 127) + 64) + 16) + 4) & 0xaa) >> 1) | (((((((0) + 127) + 64) + 16) + 4) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((((((0) + 127) + 64) + 16) + 4) + 1) & 0xaa) >> 1) | ((((((((0) + 127) + 64) + 16) + 4) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((0) + 127) + 64) + 16) + 4) + 1) & 0xaa) >> 1) | ((((((((0) + 127) + 64) + 16) + 4) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((((((((0) + 127) + 64) + 16) + 4) + 1) & 0xaa) >> 1) | ((((((((0) + 127) + 64) + 16) + 4) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((0) + 127) + 64) + 16) + 4) + 1) & 0xaa) >> 1) | (((((((((0) + 127) + 64) + 16) + 4) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((((((0) + 127) + 64) + 16) + 4) + 2) & 0xaa) >> 1) | ((((((((0) + 127) + 64) + 16) + 4) + 2) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((0) + 127) + 64) + 16) + 4) + 2) & 0xaa) >> 1) | ((((((((0) + 127) + 64) + 16) + 4) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((((((0) + 127) + 64) + 16) + 4) + 2) & 0xaa) >> 1) | ((((((((0) + 127) + 64) + 16) + 4) + 2) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((0) + 127) + 64) + 16) + 4) + 2) & 0xaa) >> 1) | ((((((((0) + 127) + 64) + 16) + 4) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((((((0) + 127) + 64) + 16) + 4) + 2) + 1) & 0xaa) >> 1) | (((((((((0) + 127) + 64) + 16) + 4) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((((0) + 127) + 64) + 16) + 4) + 2) + 1) & 0xaa) >> 1) | (((((((((0) + 127) + 64) + 16) + 4) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((((((((((0) + 127) + 64) + 16) + 4) + 2) + 1) & 0xaa) >> 1) | (((((((((0) + 127) + 64) + 16) + 4) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((((0) + 127) + 64) + 16) + 4) + 2) + 1) & 0xaa) >> 1) | (((((((((0) + 127) + 64) + 16) + 4) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((((0) + 127) + 64) + 16) + 8) & 0xaa) >> 1) | (((((((0) +

127) + 64) + 16) + 8) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((0) + 127) + 64) + 16) + 8) & 0xaa) >> 1) | (((((((0) + 127) + 64) + 16) + 8) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((((((0) + 127) + 64) + 16) + 8) & 0xaa) >> 1) | (((((((0) + 127) + 64) + 16) + 8) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((0) + 127) + 64) + 16) + 8) & 0xaa) >> 1) | (((((((0) + 127) + 64) + 16) + 8) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((((((0) + 127) + 64) + 16) + 8) + 1) & 0xaa) >> 1) | ((((((((0) + 127) + 64) + 16) + 8) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((0) + 127) + 64) + 16) + 8) + 1) & 0xaa) >> 1) | ((((((((0) + 127) + 64) + 16) + 8) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((((0) + 127) + 64) + 16) + 8) + 1) & 0xaa) >> 1) | ((((((((0) + 127) + 64) + 16) + 8) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((0) + 127) + 64) + 16) + 8) + 1) & 0xaa) >> 1) | ((((((((0) + 127) + 64) + 16) + 8) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((((((0) + 127) + 64) + 16) + 8) + 2) & 0xaa) >> 1) | ((((((((0) + 127) + 64) + 16) + 8) + 2) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((0) + 127) + 64) + 16) + 8) + 2) & 0xaa) >> 1) | ((((((((0) + 127) + 64) + 16) + 8) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((((0) + 127) + 64) + 16) + 8) + 2) & 0xaa) >> 1) | ((((((((0) + 127) + 64) + 16) + 8) + 2) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((0) + 127) + 64) + 16) + 8) + 2) & 0xaa) >> 1) | ((((((((0) + 127) + 64) + 16) + 8) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((((((0) + 127) + 64) + 16) + 8) + 2) + 1) & 0xaa) >> 1) | (((((((((0) + 127) + 64) + 16) + 8) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((((0) + 127) + 64) + 16) + 8) + 2) + 1) & 0xaa) >> 1) | (((((((((0) + 127) + 64) + 16) + 8) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((((((0) + 127) + 64) + 16) + 8) + 2) + 1) & 0xaa) >> 1) | (((((((((0) + 127) + 64) + 16) + 8) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((((0) + 127) + 64) + 16) + 8) + 2) + 1) & 0xaa) >> 1) | ((((((((((0) + 127) + 64) + 16) + 8) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((((((0) + 127) + 64) + 16) + 8) + 4) & 0xaa) >> 1) | ((((((((0) + 127) + 64) + 16) + 8) + 4) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((((0) + 127) + 64) + 16) + 8) + 4) & 0xaa) >> 1) | ((((((((0) + 127) + 64) + 16) + 8) + 4) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((((0) + 127) + 64) + 16) + 8) + 4) & 0xaa) >> 1) | ((((((((0) + 127) + 64) + 16) + 8) + 4) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((((0) + 127) + 64) + 16) + 8) + 4) & 0xaa) >> 1) | ((((((((0) + 127) + 64) + 16) + 8) + 4) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((((((0) + 127) + 64) + 16) + 8) + 4) + 1) & 0xaa) >> 1) | (((((((((0) + 127) + 64) + 16) + 8) + 4) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((((0) + 127) + 64) + 16) + 8) + 4) + 1) & 0xaa) >> 1) | (((((((((0) + 127) + 64) + 16) + 8) + 4) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((((((0) + 127) + 64) + 16) + 8) + 4) + 1) & 0xaa) >> 1) | (((((((((0) + 127) + 64) + 16) + 8) + 4) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((((0) + 127) + 64) + 16) + 8) + 4) + 1) & 0xaa) >> 1) | (((((((((0) + 127) + 64) + 16) + 8) + 4) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((((((0) + 127) + 64) + 16) + 8) + 4) + 2) & 0xaa) >> 1) | (((((((((0) + 127) + 64) + 16) + 8) + 4) + 2) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((((0) + 127) + 64) + 16) + 8) + 4) + 2) & 0xaa) >> 1) | (((((((((0) + 127) + 64) + 16) + 8) + 4) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((((((0) + 127) + 64) + 16) + 8) + 4) + 2) & 0xaa) >> 1) | (((((((((0) + 127) + 64) + 16) + 8) + 4) + 2) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((((0) + 127) + 64) + 16) + 8) + 4) + 2) & 0xaa) >> 1) | (((((((((0) + 127) + 64) + 16) + 8) + 4) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((((((((0) + 127) + 64) + 16) + 8) + 4) + 2) + 1) & 0xaa) >> 1) | ((((((((((0) + 127) + 64) + 16) + 8) + 4) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((((((0) + 127) + 64) + 16) + 8) + 4) + 2) + 1) & 0xaa) >> 1) | ((((((((((0) + 127) + 64) + 16) + 8) + 4) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((((((((((0) + 127) + 64) + 16) + 8) + 4) + 2) + 1) & 0xaa) >> 1) | ((((((((((0) + 127) + 64) + 16) + 8) + 4) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((((((0) + 127) + 64) + 16) + 8) + 4) + 2) + 1) & 0xaa) >> 1) | ((((((((((0) + 127) + 64) + 16) + 8) + 4) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((((0) + 127) + 64) + 32) & 0xaa) >> 1) | ((((((0) + 127) + 64) + 32) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((0) + 127) + 64) + 32) & 0xaa) >> 1) | ((((((0) + 127) + 64) + 32) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((0) + 127) + 64) + 32) & 0xaa) >> 1) | ((((((0) + 127) + 64) + 32) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((0) + 127) + 64) + 32) & 0xaa) >> 1) | ((((((0) + 127) + 64) + 32) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((((0) + 127) + 64) + 32) + 1) & 0xaa) >> 1) | (((((((0) + 127) + 64) + 32) + 1) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((0) + 127) + 64) + 32) + 1) & 0xaa) >> 1) | (((((((0) + 127) + 64) + 32) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((((((0) + 127) + 64) + 32) + 1) & 0xaa) >> 1) | (((((((0) + 127) + 64) + 32) + 1) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((0) + 127) + 64) + 32) + 1) & 0xaa) >> 1) | (((((((0) + 127) + 64) + 32) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((((0) + 127) + 64) + 32) + 2) & 0xaa) >> 1) | (((((((0) + 127) + 64) + 32) + 2) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((0) + 127) + 64) + 32) + 2) & 0xaa) >> 1) | (((((((0) + 127) + 64) + 32) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((((((0) + 127) + 64

) + 32) + 2) & 0xaa) >> 1) | (((((((0) + 127) + 64) + 32) + 2) & 0x55) << 1)) & 0xcc) >> 2) | ((((((
(((0) + 127) + 64) + 32) + 2) & 0xaa) >> 1) | (((((((0) + 127) + 64) + 32) + 2) & 0x55) << 1)) &
0x33) << 2)) & 0x0f) << 4)), ((((((((((
(((((0) + 127) + 64) + 32) + 2) + 1) & 0xaa) >> 1) | (((((((0
) + 127) + 64) + 32) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((
(((((0) + 127) + 64) + 32) + 2) + 1) & 0xaa) >> 1) | (((((((0) + 127) + 64) + 32) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >>
4) | (((((((((((((0) + 127) + 64) + 32) + 2) + 1) & 0xaa) >> 1) | (((((((0) + 127) + 64) + 32) + 2)
+ 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((
(((((0) + 127) + 64) + 32) + 2) + 1) & 0xaa) >> 1) | ((((((
((0) + 127) + 64) + 32) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((
((((0) + 127) + 64) + 32) + 4) & 0xaa) >> 1) | (((((((0) + 127) + 64) + 32) + 4) & 0x55) << 1)) & 0xcc) >> 2) |
((((((
((((0) + 127) + 64) + 32) + 4) & 0xaa) >> 1) | (((((((0) + 127) + 64) + 32) + 4) & 0x55) <<
1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((((((0) + 127) + 64) + 32) + 4) & 0xaa) >> 1) | (((((((0) +
127) + 64) + 32) + 4) & 0x55) << 1)) & 0xcc) >> 2) | ((((((
(((((0) + 127) + 64) + 32) + 4) & 0xaa) >>
1) | (((((((0) + 127) + 64) + 32) + 4) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((
(((((0) + 127) + 64) + 32) + 4) + 1) & 0xaa) >> 1) | (((((((0) + 127) + 64) + 32) + 4) + 1) & 0x55) << 1))
& 0xcc) >> 2) | ((((((
(((((0) + 127) + 64) + 32) + 4) + 1) & 0xaa) >> 1) | ((((((((0) + 127) + 64) + 32) + 4) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((((((0) + 127) + 64) + 32) + 4) +
1) & 0xaa) >> 1) | (((((((0) + 127) + 64) + 32) + 4) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((
(((((0) + 127) + 64) + 32) + 4) + 1) & 0xaa) >> 1) | ((((((((0) + 127) + 64) + 32) + 4) + 1) & 0x55) <<
1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((
(((((0) + 127) + 64) + 32) + 4) + 2) & 0xaa) >> 1) | ((((((
((0) + 127) + 64) + 32) + 4) + 2) & 0x55) << 1)) & 0xcc) >> 2) | ((((((
(((((0) + 127) + 64) + 32)
+ 4) + 2) & 0xaa) >> 1) | (((((((0) + 127) + 64) + 32) + 4) + 2) & 0x55) << 1)) & 0x33) << 2)) &
0xf0) >> 4) | ((((((((((((((0) + 127) + 64) + 32) + 4) + 2) & 0xaa) >> 1) | ((((((((0) + 127) + 64) + 32) + 4) + 2) & 0x55) << 1)) & 0xcc) >> 2) | ((((((
((((((0) + 127) + 64) + 32) + 4) + 2) & 0xaa) >>
1) | ((((((((0) + 127) + 64) + 32) + 4) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((
(((((0) + 127) + 64) + 32) + 4) + 2) + 1) & 0xaa) >> 1) | ((((((((0) + 127) + 64) + 32) + 4) + 2) + 1
) & 0x55) << 1)) & 0xcc) >> 2) | ((((((
((((((0) + 127) + 64) + 32) + 4) + 2) + 1) & 0xaa) >> 1) | ((((((((0) + 127) + 64) + 32) + 4) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((((0) + 127) + 64) + 32) + 4) + 2) + 1) & 0xaa) >> 1) | ((((((((0) + 127) + 64) + 32) + 4) + 2) + 1
) & 0x55) << 1)) & 0xcc) >> 2) | ((((((
((((((0) + 127) + 64) + 32) + 4) + 2) + 1) & 0xaa) >> 1) | ((((((((0) + 127) + 64) + 32) + 4) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((
((((0) + 127) + 64) + 32) + 8) & 0xaa) >> 1) | (((((((0) + 127) + 64) + 32) + 8) & 0x55) << 1)) &
0xcc) >> 2) | ((((((
((((0) + 127) + 64) + 32) + 8) & 0xaa) >> 1) | (((((((0) + 127) + 64) + 32) + 8)
& 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((((((0) + 127) + 64) + 32) + 8) & 0xaa) >> 1) |
(((((((0) + 127) + 64) + 32) + 8) & 0x55) << 1)) & 0xcc) >> 2) | ((((((
((((0) + 127) + 64) + 32) +
8) & 0xaa) >> 1) | (((((((0) + 127) + 64) + 32) + 8) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)),
((((((((((
((((0) + 127) + 64) + 32) + 8) + 1) & 0xaa) >> 1) | ((((((((0) + 127) + 64) + 32) + 8) + 1
) & 0x55) << 1)) & 0xcc) >> 2) | ((((((
((((((0) + 127) + 64) + 32) + 8) + 1) & 0xaa) >> 1) | ((((((((0
) + 127) + 64) + 32) + 8) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((((0) + 127) +
64) + 32) + 8) + 1) & 0xaa) >> 1) | ((((((((0) + 127) + 64) + 32) + 8) + 1) & 0x55) << 1)) & 0xcc)
>> 2) | ((((((
((((((0) + 127) + 64) + 32) + 8) + 1) & 0xaa) >> 1) | ((((((((0) + 127) + 64) + 32) + 8
) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((
(((((0) + 127) + 64) + 32) + 8) + 2) & 0xaa) >> 1) | ((((((((0) + 127) + 64) + 32) + 8) + 2) & 0x55) << 1)) & 0xcc) >> 2) | ((((((
((((((0) +
127) + 64) + 32) + 8) + 2) & 0xaa) >> 1) | ((((((((0) + 127) + 64) + 32) + 8) + 2) & 0x55) << 1)) &
0x33) << 2)) & 0xf0) >> 4) | ((((((((((((((0) + 127) + 64) + 32) + 8) + 2) & 0xaa) >> 1) | ((((((((0)
+ 127) + 64) + 32) + 8) + 2) & 0x55) << 1)) & 0xcc) >> 2) | ((((((
((((((0) + 127) + 64) + 32) + 8) + 2
) & 0xaa) >> 1) | ((((((((0) + 127) + 64) + 32) + 8) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) <<
4)), ((((((((((
((((((0) + 127) + 64) + 32) + 8) + 2) + 1) & 0xaa) >> 1) | (((((((((0) + 127) + 64) +
32) + 8) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((
(((((((0) + 127) + 64) + 32) + 8) + 2) + 1)
& 0xaa) >> 1) | (((((((((0) + 127) + 64) + 32) + 8) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0)
>> 4) | (((((((((((((((0) + 127) + 64) + 32) + 8) + 2) + 1) & 0xaa) >> 1) | (((((((((0) + 127) + 64)
+ 32) + 8) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((
(((((((0) + 127) + 64) + 32) + 8) + 2) + 1)
& 0xaa) >> 1) | (((((((((0) + 127) + 64) + 32) + 8) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f)
<< 4)), ((((((((((
((((((0) + 127) + 64) + 32) + 8) + 4) & 0xaa) >> 1) | ((((((((0) + 127) + 64) + 32)
+ 8) + 4) & 0x55) << 1)) & 0xcc) >> 2) | ((((((
((((((0) + 127) + 64) + 32) + 8) + 4) & 0xaa) >> 1) | (((((((0) + 127) + 64) + 32) + 8) + 4) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((((0) +
127) + 64) + 32) + 8) + 4) & 0xaa) >> 1) | ((((((((0) + 127) + 64) + 32) + 8) + 4) & 0x55) << 1)) & 0xcc) >> 2) | ((((((
((((((0) + 127) + 64) + 32) + 8) + 4) & 0xaa) >> 1) | (((((((0) + 127) + 64) +
32) + 8) + 4) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((
((((((0) + 127) + 64) + 32) + 8)

+ 4) + 1) & 0xaa) >> 1) | (((((((((0) + 127) + 64) + 32) + 8) + 4) + 1) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((((0) + 127) + 64) + 32) + 8) + 4) + 1) & 0xaa) >> 1) | ((((((((0) + 127) + 64) + 32) + 8) + 4) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((((((((0) + 127) + 64) + 32) + 8) + 4) + 1) & 0xaa) >> 1) | ((((((((0) + 127) + 64) + 32) + 8) + 4) + 1) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((((0) + 127) + 64) + 32) + 8) + 4) + 1) & 0xaa) >> 1) | ((((((((0) + 127) + 64) + 32) + 8) + 4) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((((((((0) + 127) + 64) + 32) + 8) + 4) + 2) & 0xaa) >> 1) | (((((((((0) + 127) + 64) + 32) + 8) + 4) + 2) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((((0) + 127) + 64) + 32) + 8) + 4) + 2) & 0xaa) >> 1) | ((((((((0) + 127) + 64) + 32) + 8) + 4) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((((((((0) + 127) + 64) + 32) + 8) + 4) + 2) & 0xaa) >> 1) | (((((((((0) + 127) + 64) + 32) + 8) + 4) + 2) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((((0) + 127) + 64) + 32) + 8) + 4) + 2) & 0xaa) >> 1) | (((((((((0) + 127) + 64) + 32) + 8) + 4) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((((((((0) + 127) + 64) + 32) + 8) + 4) + 2) + 1) & 0xaa) >> 1) | (((((((((0) + 127) + 64) + 32) + 8) + 4) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((((0) + 127) + 64) + 32) + 8) + 4) + 2) + 1) & 0xaa) >> 1) | (((((((((0) + 127) + 64) + 32) + 8) + 4) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((((((0) + 127) + 64) + 32) + 8) + 4) + 2) + 1) & 0xaa) >> 1) | (((((((((0) + 127) + 64) + 32) + 8) + 4) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((((0) + 127) + 64) + 32) + 8) + 4) + 2) + 1) & 0xaa) >> 1) | (((((((((0) + 127) + 64) + 32) + 8) + 4) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((((((0) + 127) + 64) + 32) + 16) & 0xaa) >> 1) | (((((((0) + 127) + 64) + 32) + 16) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((0) + 127) + 64) + 32) + 16) & 0xaa) >> 1) | (((((((0) + 127) + 64) + 32) + 16) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((((((0) + 127) + 64) + 32) + 16) & 0xaa) >> 1) | (((((((0) + 127) + 64) + 32) + 16) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((0) + 127) + 64) + 32) + 16) & 0xaa) >> 1) | (((((((0) + 127) + 64) + 32) + 16) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((((((0) + 127) + 64) + 32) + 16) + 1) & 0xaa) >> 1) | ((((((((0) + 127) + 64) + 32) + 16) + 1) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((((0) + 127) + 64) + 32) + 16) + 1) & 0xaa) >> 1) | ((((((((0) + 127) + 64) + 32) + 16) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((((((((0) + 127) + 64) + 32) + 16) + 1) & 0xaa) >> 1) | ((((((((0) + 127) + 64) + 32) + 16) + 1) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((((0) + 127) + 64) + 32) + 16) + 1) & 0xaa) >> 1) | (((((((((0) + 127) + 64) + 32) + 16) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((((((0) + 127) + 64) + 32) + 16) + 2) & 0xaa) >> 1) | ((((((((0) + 127) + 64) + 32) + 16) + 2) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((((0) + 127) + 64) + 32) + 16) + 2) & 0xaa) >> 1) | ((((((((0) + 127) + 64) + 32) + 16) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((((((((0) + 127) + 64) + 32) + 16) + 2) & 0xaa) >> 1) | ((((((((0) + 127) + 64) + 32) + 16) + 2) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((((0) + 127) + 64) + 32) + 16) + 2) & 0xaa) >> 1) | ((((((((0) + 127) + 64) + 32) + 16) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((((((((0) + 127) + 64) + 32) + 16) + 2) + 1) & 0xaa) >> 1) | (((((((((0) + 127) + 64) + 32) + 16) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((((0) + 127) + 64) + 32) + 16) + 2) + 1) & 0xaa) >> 1) | (((((((((0) + 127) + 64) + 32) + 16) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((((((0) + 127) + 64) + 32) + 16) + 2) + 1) & 0xaa) >> 1) | (((((((((0) + 127) + 64) + 32) + 16) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((((0) + 127) + 64) + 32) + 16) + 2) + 1) & 0xaa) >> 1) | (((((((((0) + 127) + 64) + 32) + 16) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((((((((0) + 127) + 64) + 32) + 16) + 4) & 0xaa) >> 1) | ((((((((0) + 127) + 64) + 32) + 16) + 4) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((((0) + 127) + 64) + 32) + 16) + 4) & 0xaa) >> 1) | ((((((((0) + 127) + 64) + 32) + 16) + 4) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((((((((0) + 127) + 64) + 32) + 16) + 4) & 0xaa) >> 1) | ((((((((0) + 127) + 64) + 32) + 16) + 4) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((((0) + 127) + 64) + 32) + 16) + 4) & 0xaa) >> 1) | ((((((((0) + 127) + 64) + 32) + 16) + 4) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((((((((0) + 127) + 64) + 32) + 16) + 4) + 1) & 0xaa) >> 1) | (((((((((0) + 127) + 64) + 32) + 16) + 4) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((((0) + 127) + 64) + 32) + 16) + 4) + 1) & 0xaa) >> 1) | (((((((((0) + 127) + 64) + 32) + 16) + 4) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((((((0) + 127) + 64) + 32) + 16) + 4) + 1) & 0xaa) >> 1) | (((((((((0) + 127) + 64) + 32) + 16) + 4) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((((0) + 127) + 64) + 32) + 16) + 4) + 1) & 0xaa) >> 1) | (((((((((0) + 127) + 64) + 32) + 16) + 4) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((((((((0) + 127) + 64) + 32) + 16) + 4) + 2) & 0xaa) >> 1) | (((((((((0) + 127) + 64) + 32) + 16) + 4) + 2) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((((0) + 127) + 64) + 32) + 16) + 4) + 2) & 0xaa) >> 1) | (((((((((0) + 127) + 64) + 32) + 16) + 4) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((((((((0) + 127) + 64) + 32) + 16) + 4) + 2) & 0xaa) >> 1) | (((((((((0) + 127) + 64) + 32) + 16) + 4) + 2) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((((0) + 127) + 64) + 32) + 16) + 4) + 2) & 0xaa) >> 1) | (((((((((0) + 127) + 64) + 32) + 16) + 4) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((((((((0) + 127) + 64) + 32) + 16) + 4) + 2) & 0xaa) >> 1) | (((((((((0) + 127) + 64) + 32) + 16) + 4) + 2) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((((0) + 127) + 64) + 32) + 16) + 4) + 2) & 0xaa) >> 1) | (((((((((0) + 127) + 64) + 32) + 16) + 4)

+ 2) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((((((((0) + 127) + 64) + 32) + 16) + 4) + 2) + 1) & 0xaa) >> 1) | (((((((((((0) + 127) + 64) + 32) + 16) + 4) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((((0) + 127) + 64) + 32) + 16) + 4) + 2) + 1) & 0xaa) >> 1) | ((((((((((((0) + 127) + 64) + 32) + 16) + 4) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((((((0) + 127) + 64) + 32) + 16) + 4) + 2) + 1) & 0xaa) >> 1) | ((((((((((((0) + 127) + 64) + 32) + 16) + 4) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((((0) + 127) + 64) + 32) + 16) + 4) + 2) + 1) & 0xaa) >> 1) | ((((((((((((0) + 127) + 64) + 32) + 16) + 4) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((((((0) + 127) + 64) + 32) + 16) + 8) & 0xaa) >> 1) | ((((((((((0) + 127) + 64) + 32) + 16) + 8) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((0) + 127) + 64) + 32) + 16) + 8) & 0xaa) >> 1) | ((((((((((0) + 127) + 64) + 32) + 16) + 8) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((((0) + 127) + 64) + 32) + 16) + 8) & 0xaa) >> 1) | ((((((((((0) + 127) + 64) + 32) + 16) + 8) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((0) + 127) + 64) + 32) + 16) + 8) & 0xaa) >> 1) | ((((((((((0) + 127) + 64) + 32) + 16) + 8) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((((((((0) + 127) + 64) + 32) + 16) + 8) + 1) & 0xaa) >> 1) | (((((((((((0) + 127) + 64) + 32) + 16) + 8) + 1) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((((0) + 127) + 64) + 32) + 16) + 8) + 1) & 0xaa) >> 1) | (((((((((((0) + 127) + 64) + 32) + 16) + 8) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((((((0) + 127) + 64) + 32) + 16) + 8) + 1) & 0xaa) >> 1) | (((((((((((0) + 127) + 64) + 32) + 16) + 8) + 1) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((((0) + 127) + 64) + 32) + 16) + 8) + 1) & 0xaa) >> 1) | (((((((((((0) + 127) + 64) + 32) + 16) + 8) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((((((((0) + 127) + 64) + 32) + 16) + 8) + 2) & 0xaa) >> 1) | (((((((((((0) + 127) + 64) + 32) + 16) + 8) + 2) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((((0) + 127) + 64) + 32) + 16) + 8) + 2) & 0xaa) >> 1) | ((((((((((0) + 127) + 64) + 32) + 16) + 8) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((((((0) + 127) + 64) + 32) + 16) + 8) + 2) & 0xaa) >> 1) | (((((((((((0) + 127) + 64) + 32) + 16) + 8) + 2) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((((0) + 127) + 64) + 32) + 16) + 8) + 2) & 0xaa) >> 1) | (((((((((0) + 127) + 64) + 32) + 16) + 8) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((((((((0) + 127) + 64) + 32) + 16) + 8) + 2) + 1) & 0xaa) >> 1) | ((((((((((((0) + 127) + 64) + 32) + 16) + 8) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((((0) + 127) + 64) + 32) + 16) + 8) + 2) + 1) & 0xaa) >> 1) | ((((((((((0) + 127) + 64) + 32) + 16) + 8) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((((((((((0) + 127) + 64) + 32) + 16) + 8) + 2) + 1) & 0xaa) >> 1) | ((((((((((((0) + 127) + 64) + 32) + 16) + 8) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((((0) + 127) + 64) + 32) + 16) + 8) + 2) + 1) & 0xaa) >> 1) | ((((((((((0) + 127) + 64) + 32) + 16) + 8) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((((((((0) + 127) + 64) + 32) + 16) + 8) + 4) & 0xaa) >> 1) | (((((((((((0) + 127) + 64) + 32) + 16) + 8) + 4) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((((0) + 127) + 64) + 32) + 16) + 8) + 4) & 0xaa) >> 1) | (((((((((0) + 127) + 64) + 32) + 16) + 8) + 4) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((((((0) + 127) + 64) + 32) + 16) + 8) + 4) & 0xaa) >> 1) | (((((((((((0) + 127) + 64) + 32) + 16) + 8) + 4) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((((0) + 127) + 64) + 32) + 16) + 8) + 4) & 0xaa) >> 1) | (((((((((0) + 127) + 64) + 32) + 16) + 8) + 4) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((((((((0) + 127) + 64) + 32) + 16) + 8) + 4) + 1) & 0xaa) >> 1) | ((((((((((((0) + 127) + 64) + 32) + 16) + 8) + 4) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((((0) + 127) + 64) + 32) + 16) + 8) + 4) + 1) & 0xaa) >> 1) | ((((((((((0) + 127) + 64) + 32) + 16) + 8) + 4) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((((((((((0) + 127) + 64) + 32) + 16) + 8) + 4) + 1) & 0xaa) >> 1) | ((((((((((((0) + 127) + 64) + 32) + 16) + 8) + 4) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((((0) + 127) + 64) + 32) + 16) + 8) + 4) + 1) & 0xaa) >> 1) | ((((((((((0) + 127) + 64) + 32) + 16) + 8) + 4) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((((((((0) + 127) + 64) + 32) + 16) + 8) + 4) + 2) & 0xaa) >> 1) | ((((((((((((0) + 127) + 64) + 32) + 16) + 8) + 4) + 2) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((((0) + 127) + 64) + 32) + 16) + 8) + 4) + 2) & 0xaa) >> 1) | (((((((((0) + 127) + 64) + 32) + 16) + 8) + 4) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((((((((((0) + 127) + 64) + 32) + 16) + 8) + 4) + 2) & 0xaa) >> 1) | ((((((((((((0) + 127) + 64) + 32) + 16) + 8) + 4) + 2) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((((0) + 127) + 64) + 32) + 16) + 8) + 4) + 2) & 0xaa) >> 1) | (((((((((0) + 127) + 64) + 32) + 16) + 8) + 4) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((((((((((0) + 127) + 64) + 32) + 16) + 8) + 4) + 2) + 1) & 0xaa) >> 1) | (((((((((((((0) + 127) + 64) + 32) + 16) + 8) + 4) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((((((0) + 127) + 64) + 32) + 16) + 8) + 4) + 2) + 1) & 0xaa) >> 1) | (((((((((((0) + 127) + 64) + 32) + 16) + 8) + 4) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((((((((((((0) + 127) + 64) + 32) + 16) + 8) + 4) + 2) + 1) & 0xaa) >> 1) | ((((((((((((((0) + 127) + 64) + 32) + 16) + 8) + 4) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((((((((0) + 127) + 64) + 32) + 16) + 8) + 4) + 2) + 1) & 0xaa) >> 1) | ((((((((((((0) + 127) + 64) + 32) + 16) + 8) + 4) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), }

8.30.1 Macro Definition Documentation

8.30.1.1 #define BR1(x) ((((x) & 0xaa) >> 1) | (((x) & 0x55) << 1))

Definition at line 9 of file led_keyboard.c.

8.30.1.2 #define BR2(x) (((BR1(x) & 0xcc) >> 2) | ((BR1(x) & 0x33) << 2))

Definition at line 10 of file led_keyboard.c.

8.30.1.3 #define BR4(x) (((BR2(x) & 0xf0) >> 4) | ((BR2(x) & 0x0f) << 4))

Definition at line 11 of file led_keyboard.c.

8.30.1.4 #define O0(i) BR4(i),

Definition at line 12 of file led_keyboard.c.

8.30.1.5 #define O1(i) O0(i) O0((i) + 1)

Definition at line 13 of file led_keyboard.c.

8.30.1.6 #define O2(i) O1(i) O1((i) + 2)

Definition at line 14 of file led_keyboard.c.

8.30.1.7 #define O3(i) O2(i) O2((i) + 4)

Definition at line 15 of file led_keyboard.c.

8.30.1.8 #define O4(i) O3(i) O3((i) + 8)

Definition at line 16 of file led_keyboard.c.

8.30.1.9 #define O5(i) O4(i) O4((i) + 16)

Definition at line 17 of file led_keyboard.c.

8.30.1.10 #define O6(i) O5(i) O5((i) + 32)

Definition at line 18 of file led_keyboard.c.

8.30.1.11 #define O7(i) O6(i) O6((i) + 64)

Definition at line 19 of file led_keyboard.c.

8.30.1.12 #define O8(i) O7(i) O7((i) + 127)

Definition at line 20 of file led_keyboard.c.

8.30.2 Function Documentation

8.30.2.1 int loadrgb_kb (usbdevice * kb, lighting * light, int mode)

Since Firmware Version 2.05 the answers for getting the stored color-maps from the hardware has changed a bit. So comparing for the correct answer cannot validate against the cmd, and has to be done against a third map.

Definition at line 181 of file led_keyboard.c.

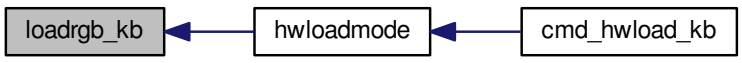
References lighting::b, ckb_err, usbdevice::fwversion, lighting::g, MSG_SIZE, N_KEYS_HW, lighting::r, usbrecv, and usbsend.

Referenced by hwloadmode().

```
181                                                     {
182     if(kb->fwversion >= 0x0120){
183         uchar data_pkt[12][MSG_SIZE] = {
184             { 0x0e, 0x14, 0x03, 0x01, 0x01, mode + 1, 0x01 },
185             { 0xff, 0x01, 60, 0 },
186             { 0xff, 0x02, 60, 0 },
187             { 0xff, 0x03, 24, 0 },
188             { 0x0e, 0x14, 0x03, 0x01, 0x01, mode + 1, 0x02 },
189             { 0xff, 0x01, 60, 0 },
190             { 0xff, 0x02, 60, 0 },
191             { 0xff, 0x03, 24, 0 },
192             { 0x0e, 0x14, 0x03, 0x01, 0x01, mode + 1, 0x03 },
193             { 0xff, 0x01, 60, 0 },
194             { 0xff, 0x02, 60, 0 },
195             { 0xff, 0x03, 24, 0 },
196         };
197         uchar in_pkt[4][MSG_SIZE] = {
198             { 0x0e, 0x14, 0x03, 0x01 },
199             { 0xff, 0x01, 60, 0 },
200             { 0xff, 0x02, 60, 0 },
201             { 0xff, 0x03, 24, 0 },
202         };
203
207
208         uchar cmp_pkt[4][4] = {
209             { 0x0e, 0x14, 0x03, 0x01 },
210             { 0x0e, 0xff, 0x01, 60 },
211             { 0x0e, 0xff, 0x02, 60 },
212             { 0x0e, 0xff, 0x03, 24 },
213         };
214         // Read colors
215         uchar* colors[3] = { light->r, light->g, light->b };
216         for(int clr = 0; clr < 3; clr++){
217             for(int i = 0; i < 4; i++){
218                 if(!usbrecv(kb, data_pkt[i + clr * 4], in_pkt[i]))
219                     return -1;
220                 // Make sure the first four bytes match
221                 // see comment above
222                 // if(memcmp(p, data_pkt[i + clr * 4], 4)){
223                 if (memcmp(in_pkt[i], (kb->fwversion >= 0x0205)? cmp_pkt[i] : data_pkt[i + clr * 4
    ], 4)) {
224                     ckb_err("Bad input header\n");
225                     ckb_err("color = %d, i = %d, mode = %d\nInput(Antwort): %2.2x %2.2x %2.2x %2.2x
     %2.2x %2.2x %2.2x %2.2x\nOutput (Frage): %2.2x %2.2x %2.2x %2.2x\n", clr, i, mode,
226                         in_pkt[i][0], in_pkt[i][1], in_pkt[i][2], in_pkt[i][3], in_pkt[i][4], in_pkt[i][5],
     in_pkt[i][6], in_pkt[i][7],
227                             // data_pkt[i + clr * 4][0],    data_pkt[i + clr * 4 ][1],  data_pkt[i + clr *
     4 ][2],  data_pkt[i + clr * 4 ][3]);
228                         cmp_pkt[i][0], cmp_pkt[i][1], cmp_pkt[i][2], cmp_pkt[i][3]);
229                     in_pkt[2][0] = 0x99;
230                     in_pkt[2][1] = 0x99;
231                     in_pkt[2][2] = 0x99;
232                     in_pkt[2][3] = 0x99;
233                     usbrecv(kb, in_pkt[2], in_pkt[2]); // just to find it in the wireshark log
234                     return -1;
235                 }
236             }
237             // Copy colors to lighting. in_pkt[0] is irrelevant.
238             memcpy(colors[clr], in_pkt[1] + 4, 60);
239             memcpy(colors[clr] + 60, in_pkt[2] + 4, 60);
240             memcpy(colors[clr] + 120, in_pkt[3] + 4, 24);
241         }
242     } else {
243         uchar data_pkt[5][MSG_SIZE] = {
244             { 0x0e, 0x14, 0x02, 0x01, 0x01, mode + 1, 0 },
245             { 0xff, 0x01, 60, 0 },
246             { 0xff, 0x02, 60, 0 },
247             { 0xff, 0x03, 60, 0 },
248             { 0xff, 0x04, 36, 0 },
```

```
249         };
250         uchar in_pkt[4][MSG_SIZE] = {
251             { 0xff, 0x01, 60, 0 },
252             { 0xff, 0x02, 60, 0 },
253             { 0xff, 0x03, 60, 0 },
254             { 0xff, 0x04, 36, 0 },
255         };
256         // Write initial packet
257         if(!usbsend(kb, data_pkt[0], 1))
258             return -1;
259         // Read colors
260         for(int i = 1; i < 5; i++){
261             if(!usbrecv(kb, data_pkt[i],in_pkt[i - 1]))
262                 return -1;
263             if(memcmp(in_pkt[i - 1], data_pkt[i], 4)){
264                 ckb_err("Bad input header\n");
265                 return -1;
266             }
267         }
268         // Copy the data back to the mode
269         uint8_t mr[N_KEYS_HW / 2], mg[N_KEYS_HW / 2], mb[
    N_KEYS_HW / 2];
270         memcpy(mr,      in_pkt[0] +  4, 60);
271         memcpy(mr + 60, in_pkt[1] +  4, 12);
272         memcpy(mg,      in_pkt[1] + 16, 48);
273         memcpy(mg + 48, in_pkt[2] +  4, 24);
274         memcpy(mb,      in_pkt[2] + 28, 36);
275         memcpy(mb + 36, in_pkt[3] +  4, 36);
276         // Unpack LED data to 8bpc format
277         for(int i = 0; i < N_KEYS_HW; i++){
278             int     i_2 = i / 2;
279             uint8_t r, g, b;
280
281             // 3-bit intensities stored in alternate nybbles.
282             if (i & 1) {
283                 r = 7 - (mr[i_2] >> 4);
284                 g = 7 - (mg[i_2] >> 4);
285                 b = 7 - (mb[i_2] >> 4);
286             } else {
287                 r = 7 - (mr[i_2] & 0x0F);
288                 g = 7 - (mg[i_2] & 0x0F);
289                 b = 7 - (mb[i_2] & 0x0F);
290             }
291             // Scale 3-bit values up to 8 bits.
292             light->r[i] = r << 5 | r << 2 | r >> 1;
293             light->g[i] = g << 5 | g << 2 | g >> 1;
294             light->b[i] = b << 5 | b << 2 | b >> 1;
295         }
296     }
297     return 0;
298 }
```

Here is the caller graph for this function:



8.30.2.2 static void makergb_512 (const lighting * *light*, uchar *data_pkt[5][64]*, uchar(*)(int, uchar) *ditherfn*) [static]

Definition at line 36 of file led_keyboard.c.

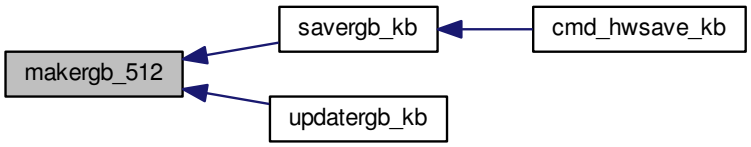
References lighting::b, lighting::g, N_KEYS_HW, and lighting::r.

Referenced by savergb_kb(), and updatergb_kb().

```
37                                                      {
38     uchar r[N_KEYS_HW / 2], g[N_KEYS_HW / 2], b[N_KEYS_HW / 2];
```

```
39      // Compress RGB values to a 512-color palette
40      for(int i = 0; i < N_KEYS_HW; i += 2){
41          char r1 = ditherfn(i, light->r[i]), r2 = ditherfn(i + 1, light->r[i + 1]);
42          char g1 = ditherfn(i, light->g[i]), g2 = ditherfn(i + 1, light->g[i + 1]);
43          char b1 = ditherfn(i, light->b[i]), b2 = ditherfn(i + 1, light->b[i + 1]);
44          r[i / 2] = (7 - r2) << 4 | (7 - r1);
45          g[i / 2] = (7 - g2) << 4 | (7 - g1);
46          b[i / 2] = (7 - b2) << 4 | (7 - b1);
47      }
48      memcpy(data_pkt[0] + 4, r, 60);
49      memcpy(data_pkt[1] + 4, r + 60, 12);
50      memcpy(data_pkt[1] + 16, g, 48);
51      memcpy(data_pkt[2] + 4, g + 48, 24);
52      memcpy(data_pkt[2] + 28, b, 36);
53      memcpy(data_pkt[3] + 4, b + 36, 36);
54  }
```

Here is the caller graph for this function:



8.30.2.3 static void makergb_full (const lighting * *light*, uchar *data_pkt[12][64]*) [static]

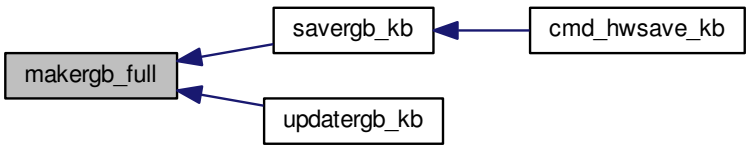
Definition at line 56 of file led_keyboard.c.

References lighting::b, lighting::g, and lighting::r.

Referenced by savergb_kb(), and updatergb_kb().

```
56                                                                                  {
57      const uchar* r = light->r, *g = light->g, *b = light->b;
58      // Red
59      memcpy(data_pkt[0] + 4, r, 60);
60      memcpy(data_pkt[1] + 4, r + 60, 60);
61      memcpy(data_pkt[2] + 4, r + 120, 24);
62      // Green (final R packet is blank)
63      memcpy(data_pkt[4] + 4, g, 60);
64      memcpy(data_pkt[5] + 4, g + 60, 60);
65      memcpy(data_pkt[6] + 4, g + 120, 24);
66      // Blue (final G packet is blank)
67      memcpy(data_pkt[8] + 4, b, 60);
68      memcpy(data_pkt[9] + 4, b + 60, 60);
69      memcpy(data_pkt[10] + 4, b + 120, 24);
70  }
```

Here is the caller graph for this function:



8.30.2.4 static uchar ordered8to3 (int *index*, uchar *value*) [static]

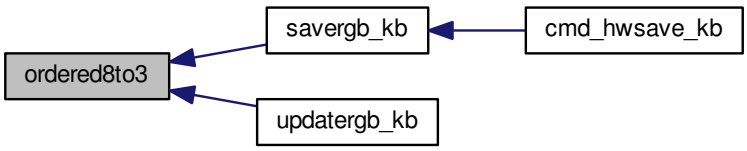
Definition at line 24 of file led_keyboard.c.

References bit_reverse_table.

Referenced by savergb_kb(), and updatergb_kb().

```
24                                                  {
25     int m = value * 7;
26     int b = m / 255;
27     if((m % 255) > bit_reverse_table[index & 0xff])
28         b++;
29     return b;
30 }
```

Here is the caller graph for this function:



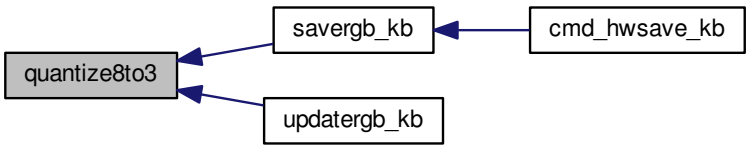
8.30.2.5 static uchar quantize8to3 (int *index*, uchar *value*) [static]

Definition at line 32 of file led_keyboard.c.

Referenced by savergb_kb(), and updatergb_kb().

```
32                                                  {
33     return value >> 5;
34 }
```

Here is the caller graph for this function:



8.30.2.6 static int rgbcmp (const lighting * *lhs*, const lighting * *rhs*) [static]

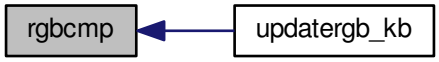
Definition at line 72 of file led_keyboard.c.

References lighting::b, lighting::g, N_KEYS_HW, and lighting::r.

Referenced by updatergb_kb().

```
72                                                      {
73     // Compare two light structures, ignore mouse zones
74     return memcmp(lhs->r, rhs->r, N_KEYS_HW) || memcmp(lhs->g, rhs->
       g, N_KEYS_HW) || memcmp(lhs->b, rhs->b, N_KEYS_HW);
75 }
```

Here is the caller graph for this function:



8.30.2.7 int savergb_kb (usbdevice * *kb,* lighting * *light,* int *mode*)

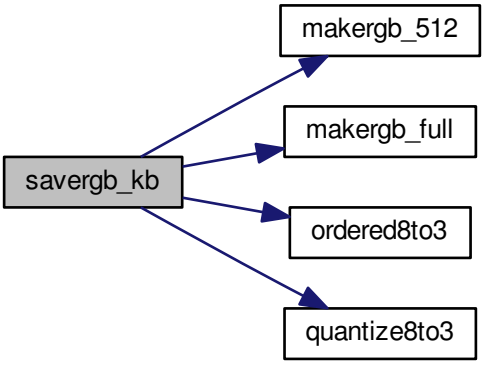
Definition at line 139 of file led_keyboard.c.

References usbdevice::dither, usbdevice::fwversion, IS_STRAFE, makergb_512(), makergb_full(), MSG_SIZE, ordered8to3(), quantize8to3(), and usbsend.

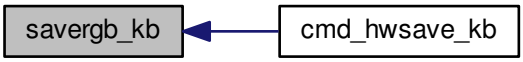
Referenced by cmd_hwsave_kb().

```
139                                                     {
140     if(kb->fwversion >= 0x0120){
141         uchar data_pkt[12][MSG_SIZE] = {
142             // Red
143             { 0x7f, 0x01, 60, 0 },
144             { 0x7f, 0x02, 60, 0 },
145             { 0x7f, 0x03, 24, 0 },
146             { 0x07, 0x14, 0x03, 0x01, 0x01, mode + 1, 0x01 },
147             // Green
148             { 0x7f, 0x01, 60, 0 },
149             { 0x7f, 0x02, 60, 0 },
150             { 0x7f, 0x03, 24, 0 },
151             { 0x07, 0x14, 0x03, 0x01, 0x01, mode + 1, 0x02 },
152             // Blue
153             { 0x7f, 0x01, 60, 0 },
154             { 0x7f, 0x02, 60, 0 },
155             { 0x7f, 0x03, 24, 0 },
156             { 0x07, 0x14, 0x03, 0x01, 0x01, mode + 1, 0x03 }
157         };
158         makergb_full(light, data_pkt);
159         if(!usbsend(kb, data_pkt[0], 12))
160             return -1;
161         if (IS_STRAFE(kb)){ // end save
162             uchar save_end_pkt[MSG_SIZE] = { 0x07, 0x14, 0x04, 0x01, 0x01 };
163             if(!usbsend(kb, save_end_pkt, 1))
164                 return -1;
165         }
166     } else {
167         uchar data_pkt[5][MSG_SIZE] = {
168             { 0x7f, 0x01, 60, 0 },
169             { 0x7f, 0x02, 60, 0 },
170             { 0x7f, 0x03, 60, 0 },
171             { 0x7f, 0x04, 36, 0 },
172             { 0x07, 0x14, 0x02, 0x00, 0x01, mode + 1 }
173         };
174         makergb_512(light, data_pkt, kb->dither ? ordered8to3 :
    quantize8to3);
175         if(!usbsend(kb, data_pkt[0], 5))
176             return -1;
177     }
178     return 0;
179 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



8.30.2.8 int updatergb_kb (usbdevice * *kb*, int *force*)

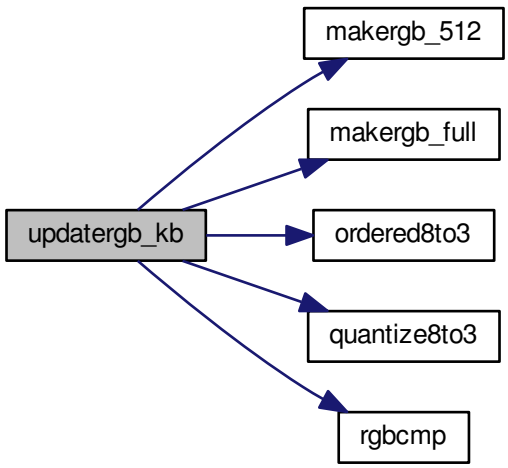
Definition at line 77 of file led_keyboard.c.

References usbdevice::active, usbprofile::currentmode, usbdevice::dither, lighting::forceupdate, IS_FULLRANGE, usbprofile::lastlight, usbmode::light, makergb_512(), makergb_full(), MSG_SIZE, ordered8to3(), usbdevice::profile, quantize8to3(), rgbcmp(), lighting::sidelight, and usbsend.

```
77                                          {
78     if(!kb->active)
79         return 0;
80     lighting* lastlight = &kb->profile->lastlight;
81     lighting* newlight = &kb->profile->currentmode->
       light;
82     // Don't do anything if the lighting hasn't changed
83     if(!force && !lastlight->forceupdate && !newlight->forceupdate
84             && !rgbcmp(lastlight, newlight) && lastlight->sidelight == newlight->
       sidelight)   // strafe sidelights
85         return 0;
86     lastlight->forceupdate = newlight->forceupdate = 0;
87
88     if(IS_FULLRANGE(kb)){
89         // Update strafe sidelights if necessary
90         if(lastlight->sidelight != newlight->sidelight) {
91             uchar data_pkt[2][MSG_SIZE] = {
92                 { 0x07, 0x05, 0x08, 0x00, 0x00 },
93                 { 0x07, 0x05, 0x02, 0, 0x03 }
94              };
95              if (newlight->sidelight)
```

```
96                  data_pkt[0][4]=1;    // turn on
97              if(!usbsend(kb, data_pkt[0], 2))
98                  return -1;
99          }
100         // 16.8M color lighting works fine on strafe and is the only way it actually works
101         uchar data_pkt[12][MSG_SIZE] = {
102             // Red
103             { 0x7f, 0x01, 0x3c, 0 },
104             { 0x7f, 0x02, 0x3c, 0 },
105             { 0x7f, 0x03, 0x18, 0 },
106             { 0x07, 0x28, 0x01, 0x03, 0x01, 0},
107             // Green
108             { 0x7f, 0x01, 0x3c, 0 },
109             { 0x7f, 0x02, 0x3c, 0 },
110             { 0x7f, 0x03, 0x18, 0 },
111             { 0x07, 0x28, 0x02, 0x03, 0x01, 0},
112             // Blue
113             { 0x7f, 0x01, 0x3c, 0 },
114             { 0x7f, 0x02, 0x3c, 0 },
115             { 0x7f, 0x03, 0x18, 0 },
116             { 0x07, 0x28, 0x03, 0x03, 0x02, 0}
117         };
118         makergb_full(newlight, data_pkt);
119         if(!usbsend(kb, data_pkt[0], 12))
120             return -1;
121     } else {
122         // On older keyboards it looks flickery and causes lighting glitches, so we don't use it.
123         uchar data_pkt[5][MSG_SIZE] = {
124             { 0x7f, 0x01, 60, 0 },
125             { 0x7f, 0x02, 60, 0 },
126             { 0x7f, 0x03, 60, 0 },
127             { 0x7f, 0x04, 36, 0 },
128             { 0x07, 0x27, 0x00, 0x00, 0xD8 }
129         };
130         makergb_512(newlight, data_pkt, kb->dither ?
    ordered8to3 : quantize8to3);
131         if(!usbsend(kb, data_pkt[0], 5))
132             return -1;
133     }
134
135     memcpy(lastlight, newlight, sizeof(lighting));
136     return 0;
137 }
```

Here is the call graph for this function:



8.30.3 Variable Documentation

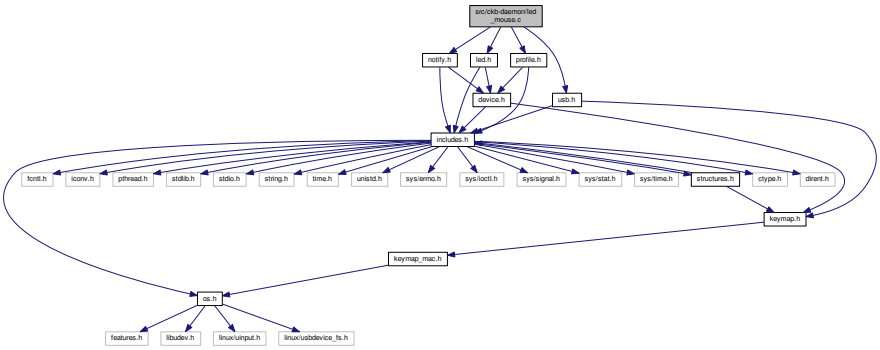
8.30.3.1 **uchar** bit_reverse_table[256] = { ((((((((((0) & 0xaa) >> 1) | (((0) & 0x55) << 1)) & 0xcc) >> 2) | ((((((0) & 0xaa) >> 1) | (((0) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((0) & 0xaa) >> 1) | (((0) & 0x55) << 1)) & 0xcc) >> 2) | ((((((0) & 0xaa) >> 1) | (((0) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((0) + 1) & 0xaa) >> 1) | ((((0) + 1) & 0x55) << 1)) & 0xcc) >> 2) | (((((((0) + 1) & 0xaa) >> 1) | ((((0) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((((0) + 1) & 0xaa) >> 1) | ((((0) + 1) & 0x55) << 1)) & 0xcc) >> 2) | (((((((0) + 1) & 0xaa) >> 1) | ((((0) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((0) + 2) & 0xaa) >> 1) | ((((0) + 2) & 0x55) << 1)) & 0xcc) >> 2) | (((((((0) + 2) & 0xaa) >> 1) | ((((0) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((((0) + 2) & 0xaa) >> 1) | ((((0) + 2) & 0x55) << 1)) & 0xcc) >> 2) | (((((((0) + 2) & 0xaa) >> 1) | ((((0) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((0) + 2) + 1) & 0xaa) >> 1) | (((((0) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((0) + 2) + 1) & 0xaa) >> 1) | (((((0) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((0) + 2) + 1) & 0xaa) >> 1) | (((((0) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((0) + 2) + 1) & 0xaa) >> 1) | (((((0) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((0) + 4) & 0xaa) >> 1) | ((((0) + 4) & 0x55) << 1)) & 0xcc) >> 2) | (((((((0) + 4) & 0xaa) >> 1) | ((((0) + 4) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((((0) + 4) & 0xaa) >> 1) | ((((0) + 4) & 0x55) << 1)) & 0xcc) >> 2) | (((((((0) + 4) & 0xaa) >> 1) | ((((0) + 4) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((0) + 4) + 1) & 0xaa) >> 1) | (((((0) + 4) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((0) + 4) + 1) & 0xaa) >> 1) | (((((0) + 4) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((0) + 4) + 1) & 0xaa) >> 1) | (((((0) + 4) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((0) + 4) + 1) & 0xaa) >> 1) | (((((0) + 4) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((0) + 4) + 2) & 0xaa) >> 1) | (((((0) + 4) + 2) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((0) + 4) + 2) & 0xaa) >> 1) | (((((0) + 4) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((0) + 4) + 2) & 0xaa) >> 1) | (((((0) + 4) + 2) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((0) + 4) + 2) & 0xaa) >> 1) | (((((0) + 4) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((((0) + 4) + 2) + 1) & 0xaa) >> 1) | ((((((0) + 4) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((0) + 4) + 2) + 1) & 0xaa) >> 1) | ((((((0) + 4) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((((((0) + 4) + 2) + 1) & 0xaa) >> 1) | ((((((0) + 4) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((0) + 4) + 2) + 1) & 0xaa) >> 1) | ((((((0) + 4) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((0) + 8) & 0xaa) >> 1) | ((((0) + 8) & 0x55) << 1)) & 0xcc) >> 2) | (((((((0) + 8) & 0xaa) >> 1) | ((((0) + 8) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((((0) + 8) & 0xaa) >> 1) | ((((0) + 8) & 0x55) << 1)) & 0xcc) >> 2) | (((((((0) + 8) & 0xaa) >> 1) | ((((0) + 8) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((0) + 8) + 1) & 0xaa) >> 1) | (((((0) + 8) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((0) + 8) + 1) & 0xaa) >> 1) | (((((0) + 8) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((0) + 8) + 1) & 0xaa) >> 1) | (((((0) + 8) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((0) + 8) + 1) & 0xaa) >> 1) | (((((0) + 8) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((0) + 8) + 2) & 0xaa) >> 1) | (((((0) + 8) + 2) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((0) + 8) + 2) & 0xaa) >> 1) | (((((0) + 8) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((0) + 8) + 2) & 0xaa) >> 1) | (((((0) + 8) + 2) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((0) + 8) + 2) & 0xaa) >> 1) | (((((0) + 8) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((((0) + 8) + 2) + 1) & 0xaa) >> 1) | ((((((0) + 8) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((0) + 8) + 2) + 1) & 0xaa) >> 1) | ((((((0) + 8) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((((((0) + 8) + 2) + 1) & 0xaa) >> 1) | ((((((0) + 8) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((0) + 8) + 2) + 1) & 0xaa) >> 1) | ((((((0) + 8) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((0) + 8) + 4) & 0xaa) >> 1) | (((((0) + 8) + 4) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((0) + 8) + 4) & 0xaa) >> 1) | (((((0) + 8) + 4) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((0) + 8) + 4) & 0xaa) >> 1) | (((((0) + 8) + 4) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((0) + 8) + 4) & 0xaa) >> 1) | (((((0) + 8) + 4) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((((0) + 8) + 4) + 1) & 0xaa) >> 1) | ((((((0) + 8) + 4) + 1) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((0) + 8) + 4) + 1) & 0xaa) >> 1) | ((((((0) + 8) + 4) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((((((0) + 8) + 4) + 1) & 0xaa) >> 1) | ((((((0) + 8) + 4) + 1) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((0) + 8) + 4) + 1) & 0xaa) >> 1) | ((((((0) + 8) + 4) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((((0) + 8) + 4) + 2) & 0xaa) >> 1) | ((((((0) + 8) + 4) + 2) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((0) + 8) + 4) + 2) & 0xaa) >> 1) | ((((((0) + 8) + 4) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((((((0) + 8) + 4) + 2) & 0xaa) >> 1) | ((((((0) + 8) + 4) + 2) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((0) + 8) + 4) + 2) & 0xaa) >> 1) | ((((((0) + 8) + 4) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((((0) + 8) + 4) + 2) + 1) & 0xaa) >> 1) | (((((((0) + 8) + 4) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((0) + 8) + 4) + 2) + 1) & 0xaa) >> 1) | (((((((0) + 8) + 4) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((((0) + 8) + 4) + 2) + 1) & 0xaa) >> 1) | (((((((0) + 8) + 4) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((0) + 8) + 4) + 2) + 1) & 0xaa) >> 1) | (((((((0) + 8) + 4) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((0) + 16) & 0xaa) >> 1) | ((((0) + 16) & 0x55) << 1)) & 0xcc) >> 2) | (((((((0) + 16) & 0xaa) >> 1) | ((((0) + 16) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((((0) + 16) & 0xaa) >> 1) | ((((0) + 16) & 0x55) << 1)) & 0xcc) >> 2) | (((((((0) + 16) & 0xaa) >> 1) | ((((0) + 16) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((0) + 16) + 1) & 0xaa) >> 1) | (((((0) + 16) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((0) + 16) + 1) & 0xaa) >> 1) | (((((0) + 16) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((0) + 16) + 1) & 0xaa) >> 1) | (((((0) + 16) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((0) + 16) + 1) & 0xaa) >> 1) | (((((0) + 16) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((0) + 16) + 2) & 0xaa) >> 1) | (((((0) + 16) + 2) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((0) + 16) + 2) & 0xaa) >> 1) | (((((0) + 16) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((0) + 16) + 2) & 0xaa) >> 1) | (((((0) + 16) + 2) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((0) + 16) + 2) & 0xaa) >> 1) | (((((0) + 16) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((((0) + 16) + 2) + 1) & 0xaa) >> 1) | ((((((0) + 16) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((0) + 16) + 2) + 1) & 0xaa) >> 1) | ((((((0) + 16) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((((((0) + 16) + 2) + 1) & 0xaa) >> 1) | ((((((0) + 16) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((0) + 16) + 2) + 1) & 0xaa) >>

Referenced by ordered8to3().

8.31 src/ckb-daemon/led_mouse.c File Reference

```
#include "led.h"
#include "notify.h"
#include "profile.h"
#include "usb.h"
```

Include dependency graph for led_mouse.c:



Functions

- static int rgbcmp (const lighting *lhs, const lighting *rhs)
- static int isblack (const usbdevice *kb, const lighting *light)
- int updatergb_mouse (usbdevice *kb, int force)
- int savergb_mouse (usbdevice *kb, lighting *light, int mode)
- int loadrgb_mouse (usbdevice *kb, lighting *light, int mode)

8.31.1 Function Documentation

8.31.1.1 static int isblack (const usbdevice * *kb*, const lighting * *light*) [static]

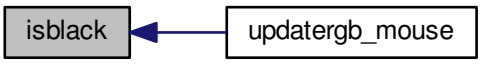
Definition at line 13 of file led_mouse.c.

References lighting::b, lighting::g, IS_M65, LED_MOUSE, N_MOUSE_ZONES, and lighting::r.

Referenced by updatergb_mouse().

```
13                                                          {
14     if(!IS_M65(kb))
15         return 0;
16     uchar black[N_MOUSE_ZONES] = { 0 };
17     return !memcmp(light->r + LED_MOUSE, black, sizeof(black)) && !memcmp(light->
       g + LED_MOUSE, black, sizeof(black)) && !memcmp(light->b + LED_MOUSE, black, sizeof(
       black));
18 }
```

Here is the caller graph for this function:



8.31.1.2 int loadrgb_mouse (usbdevice * *kb*, lighting * *light*, int *mode*)

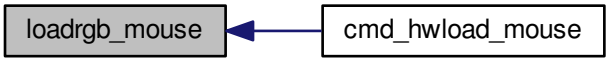
Definition at line 81 of file led_mouse.c.

References lighting::b, ckb_err, lighting::g, IS_SABRE, IS_SCIMITAR, LED_DPI, LED_MOUSE, MSG_SIZE, lighting::r, and usbrecv.

Referenced by cmd_hwload_mouse().

```
81                                                          {
82     uchar data_pkt[MSG_SIZE] = { 0x0e, 0x13, 0x10, 1, 0 };
83     uchar in_pkt[MSG_SIZE] = { 0 };
84     // Load each RGB zone
85     int zonecount = IS_SCIMITAR(kb) ? 4 : IS_SABRE(kb) ? 3 : 2;
86     for(int i = 0; i < zonecount; i++){
87         if(!usbrecv(kb, data_pkt, in_pkt))
88             return -1;
89         if(memcmp(in_pkt, data_pkt, 4)){
90             ckb_err("Bad input header\n");
91             return -2;
92         }
93         // Copy data
94         int led = LED_MOUSE + i;
95         if(led >= LED_DPI)
96             led++;          // Skip DPI light
97         light->r[led] = in_pkt[4];
98         light->g[led] = in_pkt[5];
99         light->b[led] = in_pkt[6];
100         // Set packet for next zone
101         data_pkt[2]++;
102     }
103     return 0;
104 }
```

Here is the caller graph for this function:



8.31.1.3 static int rgbcmp (const lighting * *lhs*, const lighting * *rhs*) [static]

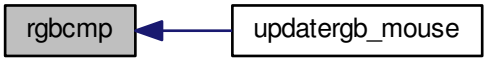
Definition at line 7 of file led_mouse.c.

References lighting::b, lighting::g, LED_MOUSE, N_MOUSE_ZONES, and lighting::r.

Referenced by updatergb_mouse().

```
7                                                                   {
8       return memcmp(lhs->r + LED_MOUSE, rhs->r + LED_MOUSE,
        N_MOUSE_ZONES) || memcmp(lhs->g + LED_MOUSE, rhs->g +
        LED_MOUSE, N_MOUSE_ZONES) || memcmp(lhs->b + LED_MOUSE, rhs->
        b + LED_MOUSE, N_MOUSE_ZONES);
9 }
```

Here is the caller graph for this function:



8.31.1.4 int savergb_mouse (usbdevice * *kb*, lighting * *light*, int *mode*)

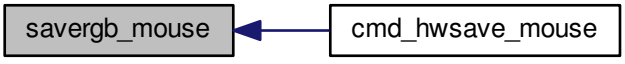
Definition at line 62 of file led_mouse.c.

References lighting::b, lighting::g, IS_SABRE, IS_SCIMITAR, LED_DPI, LED_MOUSE, MSG_SIZE, lighting::r, and usbsend.

Referenced by cmd_hwsave_mouse().

```
62                                                                  {
63      uchar data_pkt[MSG_SIZE] = { 0x07, 0x13, 0x10, 1, 0 };
64      // Save each RGB zone, minus the DPI light which is sent in the DPI packets
65      int zonecount = IS_SCIMITAR(kb) ? 4 : IS_SABRE(kb) ? 3 : 2;
66      for(int i = 0; i < zonecount; i++){
67          int led = LED_MOUSE + i;
68          if(led >= LED_DPI)
69              led++;          // Skip DPI light
70          data_pkt[4] = light->r[led];
71          data_pkt[5] = light->g[led];
72          data_pkt[6] = light->b[led];
73          if(!usbsend(kb, data_pkt, 1))
74              return -1;
75          // Set packet for next zone
76          data_pkt[2]++;
77      }
78      return 0;
79 }
```

Here is the caller graph for this function:



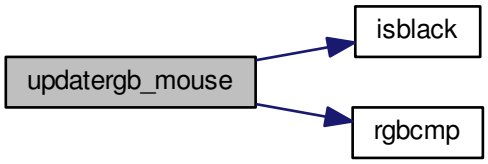
8.31.1.5 int updatergb_mouse (usbdevice * *kb*, int *force*)

Definition at line 20 of file led_mouse.c.

References usbdevice::active, lighting::b, usbprofile::currentmode, lighting::forceupdate, lighting::g, isblack(), usbprofile::lastlight, LED_MOUSE, usbmode::light, MSG_SIZE, N_MOUSE_ZONES, usbdevice::profile, lighting::r, rgbcmp(), and usbsend.

```
20                                              {
21     if(!kb->active)
22         return 0;
23     lighting* lastlight = &kb->profile->lastlight;
24     lighting* newlight = &kb->profile->currentmode->
      light;
25     // Don't do anything if the lighting hasn't changed
26     if(!force && !lastlight->forceupdate && !newlight->forceupdate
27             && !rgbcmp(lastlight, newlight))
28         return 0;
29     lastlight->forceupdate = newlight->forceupdate = 0;
30
31     // Send the RGB values for each zone to the mouse
32     uchar data_pkt[2][MSG_SIZE] = {
33         { 0x07, 0x22, N_MOUSE_ZONES, 0x01, 0 }, // RGB colors
34         { 0x07, 0x05, 0x02, 0 }                 // Lighting on/off
35     };
36     uchar* rgb_data = &data_pkt[0][4];
37     for(int i = 0; i < N_MOUSE_ZONES; i++){
38         *rgb_data++ = i + 1;
39         *rgb_data++ = newlight->r[LED_MOUSE + i];
40         *rgb_data++ = newlight->g[LED_MOUSE + i];
41         *rgb_data++ = newlight->b[LED_MOUSE + i];
42     }
43     // Send RGB data
44     if(!usbsend(kb, data_pkt[0], 1))
45         return -1;
46     int was_black = isblack(kb, lastlight), is_black = isblack(kb, newlight);
47     if(is_black){
48         // If the lighting is black, send the deactivation packet (M65 only)
49         if(!usbsend(kb, data_pkt[1], 1))
50             return -1;
51     } else if(was_black || force){
52         // If the lighting WAS black, or if we're on forced update, send the activation packet
53         data_pkt[1][4] = 1;
54         if(!usbsend(kb, data_pkt[1], 1))
55             return -1;
56     }
57
58     memcpy(lastlight, newlight, sizeof(lighting));
59     return 0;
60 }
```

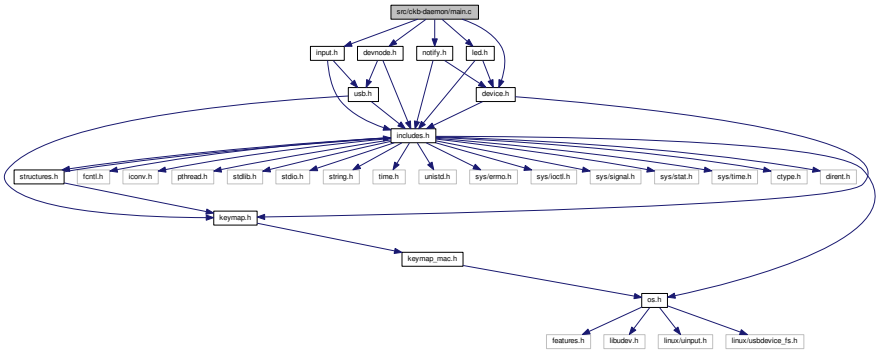
Here is the call graph for this function:



8.32 src/ckb-daemon/main.c File Reference

```
#include "device.h"
#include "devnode.h"
#include "input.h"
#include "led.h"
#include "notify.h"
```

Include dependency graph for main.c:



Functions

- static void quitWithLock (char mut)

  quitWithLock
- int restart ()
- void timespec_add (struct timespec ∗timespec, long nanoseconds)
- static void quit ()

  quit Stop working the daemon. function is called if the daemon received a sigterm In this case, locking the device-mutex is ok.
- void sighandler2 (int type)
- void sighandler (int type)
- void localecase (char ∗dst, size_t length, const char ∗src)
- int main (int argc, char ∗∗argv)

Variables

- static int main_ac
- static char ∗∗ main_av
- volatile int reset_stop
- int features_mask
- int hwload_mode

8.32.1 Function Documentation

8.32.1.1 void localecase (char ∗ *dst*, size_t *length*, const char ∗ *src*)

Definition at line 71 of file main.c.

```
71                                                          {
72     char* ldst = dst + length;
73     char s;
74     while((s = *src++)){
75         if(s == '_')
76             s = '-';
77         else
78             s = tolower(s);
79         *dst++ = s;
80         if(dst == ldst){
81             dst--;
82             break;
83         }
84     }
85     *dst = 0;
86 }
```

8.32.1.2 int main (int *argc*, char ** *argv*)

Definition at line 88 of file main.c.

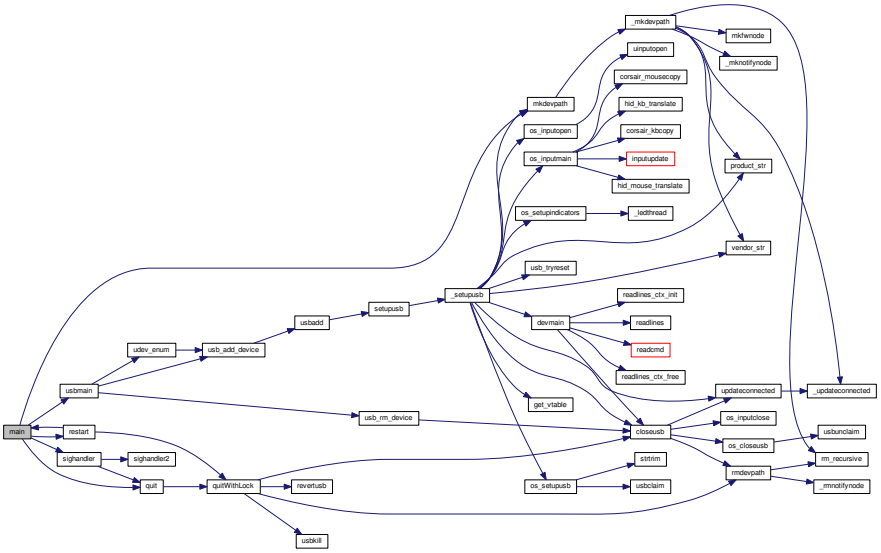
References ckb_fatal_nofile, ckb_info, ckb_info_nofile, ckb_warn_nofile, devpath, FEAT_BIND, FEAT_MOUSEACCEL, FEAT_NOTIFY, features_mask, gid, hwload_mode, keyboard, main_ac, main_av, mkdevpath(), quit(), restart(), sighandler(), and usbmain().

Referenced by restart().

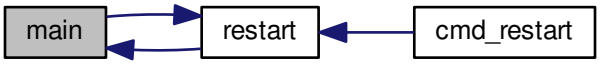
```
88                                {
89      // Set output pipes to buffer on newlines, if they weren't set that way already
90      setlinebuf(stdout);
91      setlinebuf(stderr);
92      main_ac = argc;
93      main_av = argv;
94
95      printf("    ckb: Corsair RGB driver %s\n", CKB_VERSION_STR);
96      // If --help occurs anywhere in the command-line, don't launch the program but instead print usage
97      for(int i = 1; i < argc; i++){
98          if(!strcmp(argv[i], "--help")){
99              printf(
100 #ifdef OS_MAC
101                         "Usage: ckb-daemon [--gid=<gid>] [--hwload=<always|try|never>] [--nonotify]
    [--nobind] [--nomouseaccel] [--nonroot]\n"
102 #else
103                         "Usage: ckb-daemon [--gid=<gid>] [--hwload=<always|try|never>] [--nonotify]
    [--nobind] [--nonroot]\n"
104 #endif
105                         "\n"
106                         "See https://github.com/ccMSC/ckb/blob/master/DAEMON.md for full instructions.\n"
107                         "\n"
108                         "Command-line parameters:\n"
109                         "    --gid=<gid>\n"
110                         "        Restrict access to %s* nodes to users in group <gid>.\n"
111                         "        (Ordinarily they are accessible to anyone)\n"
112                         "    --hwload=<always|try|never>\n"
113                         "        --hwload=always will force loading of stored hardware profiles on
    compatible devices. May result in long start up times.\n"
114                         "        --hwload=try will try to load the profiles, but give up if not immediately
    successful (default).\n"
115                         "        --hwload=never will ignore hardware profiles completely.\n"
116                         "    --nonotify\n"
117                         "        Disables key monitoring/notifications.\n"
118                         "        Note that this makes reactive lighting impossible.\n"
119                         "    --nobind\n"
120                         "        Disables all key rebinding, macros, and notifications. Implies --nonotify.
    \n"
121 #ifdef OS_MAC
122                         "    --nomouseaccel\n"
123                         "        Disables mouse acceleration, even if the system preferences enable it.\n"
124 #endif
125                         "    --nonroot\n"
126                         "        Allows running ckb-daemon as a non root user.\n"
127                         "        This will almost certainly not work. Use only if you know what you're
    doing.\n"
128                         "\n", devpath);
129             exit(0);
130         }
131     }
132
133     // Check PID, quit if already running
134     char pidpath[strlen(devpath) + 6];
135     snprintf(pidpath, sizeof(pidpath), "%s0/pid", devpath);
136     FILE* pidfile = fopen(pidpath, "r");
137     if(pidfile){
138         pid_t pid;
139         fscanf(pidfile, "%d", &pid);
140         fclose(pidfile);
141         if(pid > 0){
142             // kill -s 0 checks if the PID is active but doesn't send a signal
143             if(!kill(pid, 0)){
144                 ckb_fatal_nofile("ckb-daemon is already running (PID %d). Try `killall
    ckb-daemon`.\n", pid);
145                 ckb_fatal_nofile("(If you're certain the process is dead, delete %s and try
    again)\n", pidpath);
146                 return 0;
147             }
148         }
149     }
150
151     // Read parameters
152     int forceroot = 1;
153     for(int i = 1; i < argc; i++){
```

```
154         char* argument = argv[i];
155         unsigned newgid;
156         char hwload[7];
157         if(sscanf(argument, "--gid=%u", &newgid) == 1){
158             // Set dev node GID
159             gid = newgid;
160             ckb_info_nofile("Setting /dev node gid: %u\n", newgid);
161         } else if(!strcmp(argument, "--nobind")){
162             // Disable key notifications and rebinding
163             features_mask &= ~FEAT_BIND & ~FEAT_NOTIFY;
164             ckb_info_nofile("Key binding and key notifications are disabled\n");
165         } else if(!strcmp(argument, "--nonotify")){
166             // Disable key notifications
167             features_mask &= ~FEAT_NOTIFY;
168             ckb_info_nofile("Key notifications are disabled\n");
169         } else if(sscanf(argument, "--hwload=%6s", hwload) == 1){
170             if(!strcmp(hwload, "always") || !strcmp(hwload, "yes") || !strcmp(hwload, "y") || !strcmp(
    hwload, "a")){
171                 hwload_mode = 2;
172                 ckb_info_nofile("Setting hardware load: always\n");
173             } else if(!strcmp(hwload, "tryonce") || !strcmp(hwload, "try") || !strcmp(hwload, "once") || !
    strcmp(hwload, "t") || !strcmp(hwload, "o")){
174                 hwload_mode = 1;
175                 ckb_info_nofile("Setting hardware load: tryonce\n");
176             } else if(!strcmp(hwload, "never") || !strcmp(hwload, "none") || !strcmp(hwload, "no") || !
    strcmp(hwload, "n")){
177                 hwload_mode = 0;
178                 ckb_info_nofile("Setting hardware load: never\n");
179             }
180         } else if(!strcmp(argument, "--nonroot")){
181             // Allow running as a non-root user
182             forceroot = 0;
183         }
184 #ifdef OS_MAC
185         else if(!strcmp(argument, "--nomouseaccel")){
186             // On OSX, provide an option to disable mouse acceleration
187             features_mask &= ~FEAT_MOUSEACCEL;
188             ckb_info_nofile("Mouse acceleration disabled\n");
189         }
190 #endif
191     }
192
193     // Check UID
194     if(getuid() != 0){
195         if(forceroot){
196             ckb_fatal_nofile("ckb-daemon must be run as root. Try 'sudo %s'\n", argv[0]);
197             exit(0);
198         } else
199             ckb_warn_nofile("Warning: not running as root, allowing anyway per command-line
     parameter...\n");
200     }
201
202     // Make root keyboard
203     umask(0);
204     memset(keyboard, 0, sizeof(keyboard));
205     if(!mkdevpath(keyboard))
206         ckb_info("Root controller ready at %s0\n", devpath);
207
208     // Set signals
209     sigset_t signals;
210     sigfillset(&signals);
211     sigdelset(&signals, SIGTERM);
212     sigdelset(&signals, SIGINT);
213     sigdelset(&signals, SIGQUIT);
214     sigdelset(&signals, SIGUSR1);
215     // Set up signal handlers for quitting the service.
216     sigprocmask(SIG_SETMASK, &signals, 0);
217     signal(SIGTERM, sighandler);
218     signal(SIGINT, sighandler);
219     signal(SIGQUIT, sighandler);
220     signal(SIGUSR1, (void (*)())restart);
221
222     // Start the USB system
223     int result = usbmain();
224     quit();
225     return result;
226 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



8.32.1.3 static void quit ( ) [static]

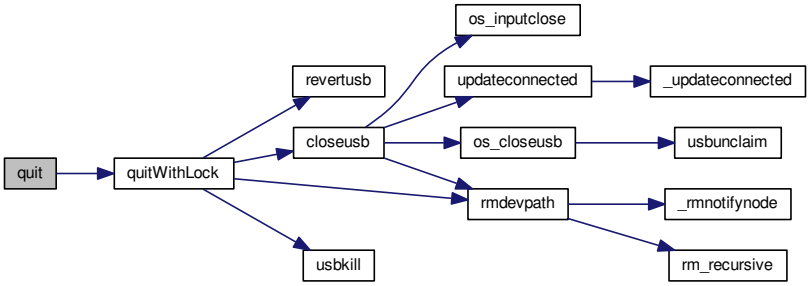
Definition at line 30 of file main.c.

References quitWithLock().

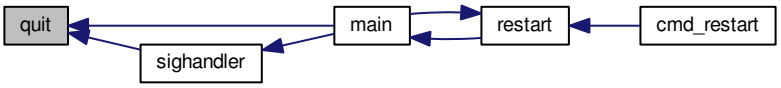
Referenced by main(), and sighandler().

```
30                  {
31      quitWithLock(1);
32  }
```

Here is the call graph for this function:



Here is the caller graph for this function:



8.32.1.4 void quitWithLock (char *mut*) [static]

Parameters

mut	try to close files maybe without locking the mutex if mut == true then lock

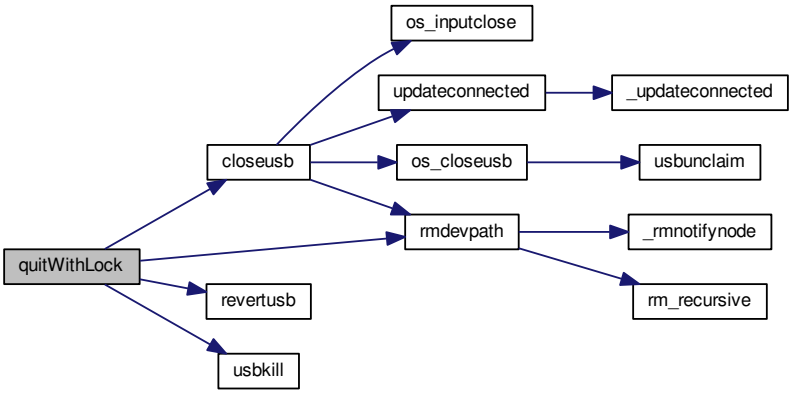
Definition at line 40 of file main.c.

References ckb_info, closeusb(), DEV_MAX, devmutex, IS_CONNECTED, keyboard, reset_stop, revertusb(), rmdevpath(), and usbkill().

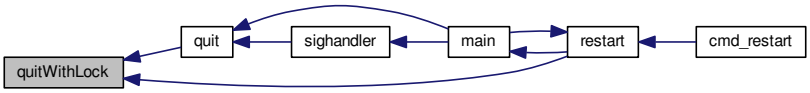
Referenced by quit(), and restart().

```
40                          {
41     // Abort any USB resets in progress
42     reset_stop = 1;
43     for(int i = 1; i < DEV_MAX; i++){
44         // Before closing, set all keyboards back to HID input mode so that the stock driver can still talk
       to them
45         if (mut) pthread_mutex_lock(devmutex + i);
46         if(IS_CONNECTED(keyboard + i)){
47             revertusb(keyboard + i);
48             closeusb(keyboard + i);
49         }
50         pthread_mutex_unlock(devmutex + i);
51     }
52     ckb_info("Closing root controller\n");
53     rmdevpath(keyboard);
54     usbkill();
55 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



8.32.1.5 int restart ( )

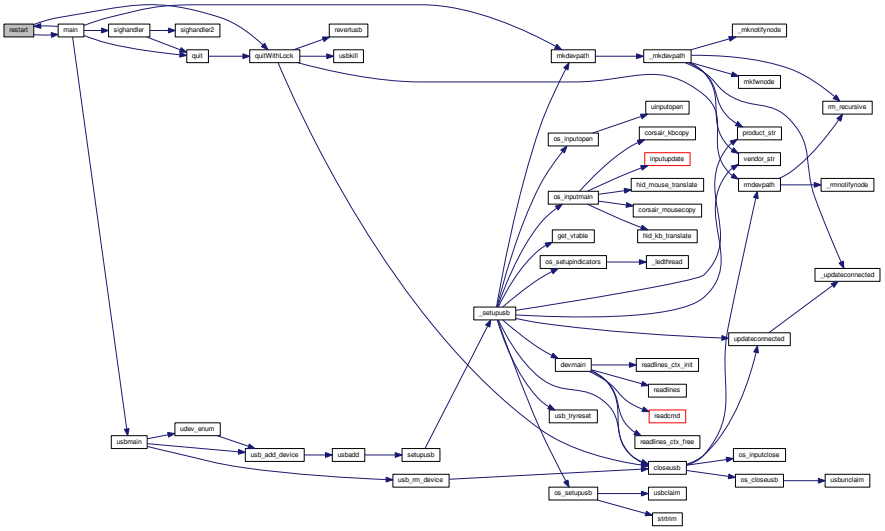
Definition at line 228 of file main.c.

References ckb_err, main(), main_ac, main_av, and quitWithLock().

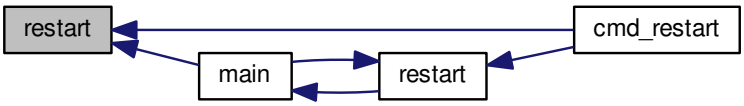
Referenced by cmd_restart(), and main().

```
228             {
229     ckb_err("restart called, running quit without mutex-lock.\n");
230     quitWithLock(0);
231     return main(main_ac, main_av);
232 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



8.32.1.6 void sighandler (int *type*)

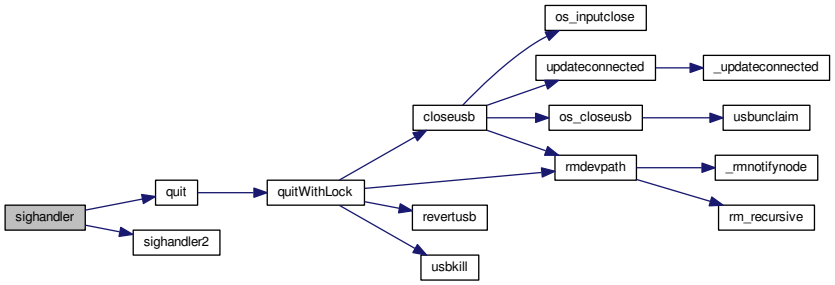
Definition at line 62 of file main.c.

References quit(), and sighandler2().

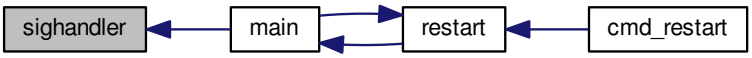
Referenced by main().

```
62                          {
63     signal(SIGTERM, sighandler2);
64     signal(SIGINT, sighandler2);
65     signal(SIGQUIT, sighandler2);
66     printf("\n[I] Caught signal %d\n", type);
67     quit();
68     exit(0);
69 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



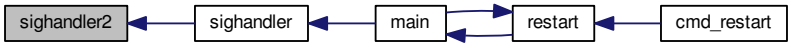
8.32.1.7 void sighandler2 (int *type*)

Definition at line 57 of file main.c.

Referenced by sighandler().

```
57                          {
58     // Don't use ckb_warn, we want an extra \n at the beginning
59     printf("\n[W] Ignoring signal %d (already shutting down)\n", type);
60 }
```

Here is the caller graph for this function:



8.32.1.8 void timespec_add (struct timespec * *timespec*, long *nanoseconds*)

Definition at line 19 of file main.c.

```
19                                                          {
20     nanoseconds += timespec->tv_nsec;
21     timespec->tv_sec += nanoseconds / 1000000000;
22     timespec->tv_nsec = nanoseconds % 1000000000;
23 }
```

8.32.2 Variable Documentation

8.32.2.1 int features_mask

Definition at line 17 of file usb.c.

Referenced by _setupusb(), and main().

8.32.2.2 int hwload_mode

Definition at line 7 of file device.c.

Referenced by main().

8.32.2.3 int main_ac [static]

Definition at line 7 of file main.c.

Referenced by main(), and restart().

8.32.2.4 char** main_av [static]

Definition at line 8 of file main.c.

Referenced by main(), and restart().

8.32.2.5 volatile int reset_stop

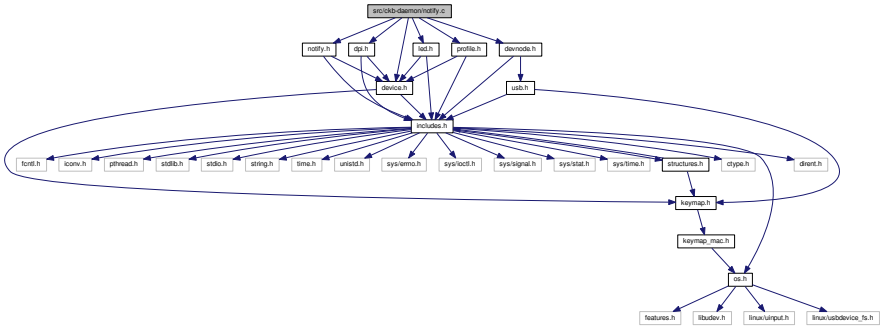
Definition at line 14 of file usb.c.

Referenced by _usbrecv(), _usbsend(), quitWithLock(), and usb_tryreset().

8.33 src/ckb-daemon/notify.c File Reference

```
#include "device.h"
#include "devnode.h"
#include "dpi.h"
#include "led.h"
#include "notify.h"
#include "profile.h"
```

Include dependency graph for notify.c:



Macros

- #define HWMODE_OR_RETURN(kb, index)
- #define HW_STANDARD

Functions

- void nprintf (usbdevice *kb, int nodenumber, usbmode *mode, const char *format,...)
- void nprintkey (usbdevice *kb, int nnumber, int keyindex, int down)
- void nprintind (usbdevice *kb, int nnumber, int led, int on)
- void cmd_notify (usbdevice *kb, usbmode *mode, int nnumber, int keyindex, const char *toggle)
- static void _cmd_get (usbdevice *kb, usbmode *mode, int nnumber, const char *setting)
- void cmd_get (usbdevice *kb, usbmode *mode, int nnumber, int dummy, const char *setting)
- int restart ()
- void cmd_restart (usbdevice *kb, usbmode *mode, int nnumber, int dummy, const char *content)

8.33.1 Macro Definition Documentation

8.33.1.1 #define HW_STANDARD

Value:

```
if(!kb->hw)                                     \
        return;                                     \
    unsigned index = INDEX_OF(mode, profile->mode); \
    /* Make sure the mode number is valid */        \
    HWMODE_OR_RETURN(kb, index)
```

Definition at line 83 of file notify.c.

Referenced by _cmd_get().

8.33.1.2 #define HWMODE_OR_RETURN(*kb, index*)

Value:

```
if(IS_K95(kb)){                 \
        if((index) >= HWMODE_K95)   \
            return;                 \
    } else {                        \
        if((index) >= HWMODE_K70)   \
            return;                 \
    }
```

Definition at line 73 of file notify.c.

8.33.2 Function Documentation

8.33.2.1 static void _cmd_get (usbdevice * *kb,* usbmode * *mode,* int *nnumber,* const char * *setting*) [static]

Definition at line 90 of file notify.c.

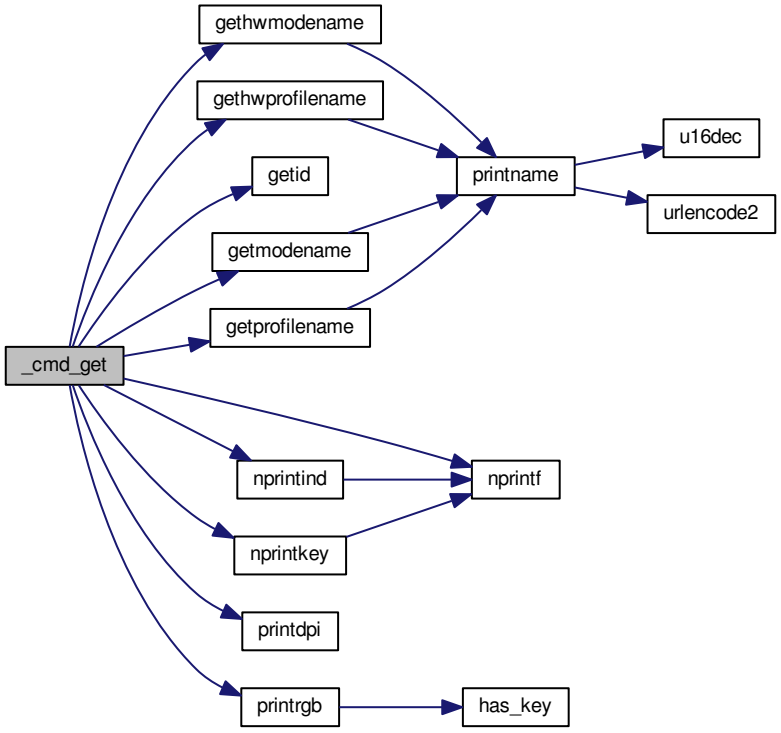
References dpiset::current, usbmode::dpi, hwprofile::dpi, gethwmodename(), gethwprofilename(), getid(), getmodename(), getprofilename(), usbdevice::hw, usbdevice::hw_ileds, HW_STANDARD, I_CAPS, I_NUM, I_SCROLL, usbmode::id, usbprofile::id, hwprofile::id, usbdevice::input, keymap, usbinput::keys, dpiset::lift, usbmode::light, hwprofile::light, usbid::modified, N_KEYS_INPUT, nprintf(), nprintind(), nprintkey(), printdpi(), printrgb(), usbdevice::profile, and dpiset::snap.

Referenced by cmd_get().

```
90                                                                                                  {
91      usbprofile* profile = kb->profile;
92      if(!strcmp(setting, ":mode")){
93          // Get the current mode number
94          nprintf(kb, nnumber, mode, "switch\n");
95          return;
96      } else if(!strcmp(setting, ":rgb")){
97          // Get the current RGB settings
98          char* rgb = printrgb(&mode->light, kb);
99          nprintf(kb, nnumber, mode, "rgb %s\n", rgb);
100          free(rgb);
101          return;
102      } else if(!strcmp(setting, ":hwrgb")){
103          // Get the current hardware RGB settings
104          HW_STANDARD;
105          char* rgb = printrgb(kb->hw->light + index, kb);
106          nprintf(kb, nnumber, mode, "hwrgb %s\n", rgb);
107          free(rgb);
108          return;
109      } else if(!strcmp(setting, ":profilename")){
110          // Get the current profile name
111          char* name = getprofilename(profile);
112          nprintf(kb, nnumber, 0, "profilename %s\n", name[0] ? name : "Unnamed");
113          free(name);
114      } else if(!strcmp(setting, ":name")){
115          // Get the current mode name
116          char* name = getmodename(mode);
117          nprintf(kb, nnumber, mode, "name %s\n", name[0] ? name : "Unnamed");
118          free(name);
119      } else if(!strcmp(setting, ":hwprofilename")){
120          // Get the current hardware profile name
121          if(!kb->hw)
122              return;
123          char* name = gethwprofilename(kb->hw);
124          nprintf(kb, nnumber, 0, "hwprofilename %s\n", name[0] ? name : "Unnamed");
125          free(name);
126      } else if(!strcmp(setting, ":hwname")){
127          // Get the current hardware mode name
128          HW_STANDARD;
129          char* name = gethwmodename(kb->hw, index);
130          nprintf(kb, nnumber, mode, "hwname %s\n", name[0] ? name : "Unnamed");
131          free(name);
132      } else if(!strcmp(setting, ":profileid")){
133          // Get the current profile ID
134          char* guid = getid(&profile->id);
135          int modified;
136          memcpy(&modified, &profile->id.modified, sizeof(modified));
137          nprintf(kb, nnumber, 0, "profileid %s %x\n", guid, modified);
138          free(guid);
139      } else if(!strcmp(setting, ":id")){
140          // Get the current mode ID
141          char* guid = getid(&mode->id);
142          int modified;
143          memcpy(&modified, &mode->id.modified, sizeof(modified));
144          nprintf(kb, nnumber, mode, "id %s %x\n", guid, modified);
145          free(guid);
146      } else if(!strcmp(setting, ":hwprofileid")){
147          // Get the current hardware profile ID
148          if(!kb->hw)
149              return;
150          char* guid = getid(&kb->hw->id[0]);
151          int modified;
152          memcpy(&modified, &kb->hw->id[0].modified, sizeof(modified));
153          nprintf(kb, nnumber, 0, "hwprofileid %s %x\n", guid, modified);
154          free(guid);
155      } else if(!strcmp(setting, ":hwid")){
156          // Get the current hardware mode ID
157          HW_STANDARD;
158          char* guid = getid(&kb->hw->id[index + 1]);
159          int modified;
160          memcpy(&modified, &kb->hw->id[index + 1].modified, sizeof(modified));
161          nprintf(kb, nnumber, mode, "hwid %s %x\n", guid, modified);
162          free(guid);
163      } else if(!strcmp(setting, ":keys")){
164          // Get the current state of all keys
165          for(int i = 0; i < N_KEYS_INPUT; i++){
166              if(!keymap[i].name)
167                  continue;
168              int byte = i / 8, bit = 1 << (i & 7);
169              uchar state = kb->input.keys[byte] & bit;
170              if(state)
171                  nprintkey(kb, nnumber, i, 1);
172          }
173      } else if(!strcmp(setting, ":i")){
174          // Get the current state of all indicator LEDs
175          if(kb->hw_ileds & I_NUM) nprintind(kb, nnumber,
    I_NUM, 1);
```

```
176         if(kb->hw_ileds & I_CAPS) nprintind(kb, nnumber,
    I_CAPS, 1);
177         if(kb->hw_ileds & I_SCROLL) nprintind(kb, nnumber,
    I_SCROLL, 1);
178     } else if(!strcmp(setting, ":dpi")){
179         // Get the current DPI levels
180         char* dpi = printdpi(&mode->dpi, kb);
181         nprintf(kb, nnumber, mode, "dpi %s\n", dpi);
182         free(dpi);
183         return;
184     } else if(!strcmp(setting, ":hwdpi")){
185         // Get the current hardware DPI levels
186         HW_STANDARD;
187         char* dpi = printdpi(kb->hw->dpi + index, kb);
188         nprintf(kb, nnumber, mode, "hwdpi %s\n", dpi);
189         free(dpi);
190         return;
191     } else if(!strcmp(setting, ":dpisel")){
192         // Get the currently-selected DPI
193         nprintf(kb, nnumber, mode, "dpisel %d\n", mode->dpi.current);
194     } else if(!strcmp(setting, ":hwdpisel")){
195         // Get the currently-selected hardware DPI
196         HW_STANDARD;
197         nprintf(kb, nnumber, mode, "hwdpisel %d\n", kb->hw->dpi[index].
    current);
198     } else if(!strcmp(setting, ":lift")){
199         // Get the mouse lift height
200         nprintf(kb, nnumber, mode, "lift %d\n", mode->dpi.lift);
201     } else if(!strcmp(setting, ":hwlift")){
202         // Get the hardware lift height
203         HW_STANDARD;
204         nprintf(kb, nnumber, mode, "hwlift %d\n", kb->hw->dpi[index].
    lift);
205     } else if(!strcmp(setting, ":snap")){
206         // Get the angle snap status
207         nprintf(kb, nnumber, mode, "snap %s\n", mode->dpi.snap ? "on" : "off");
208     } else if(!strcmp(setting, ":hwsnap")){
209         // Get the hardware angle snap status
210         HW_STANDARD;
211         nprintf(kb, nnumber, mode, "hwsnap %s\n", kb->hw->dpi[index].
    snap ? "on" : "off");
212     }
213 }
```

Here is the call graph for this function:



Here is the caller graph for this function:

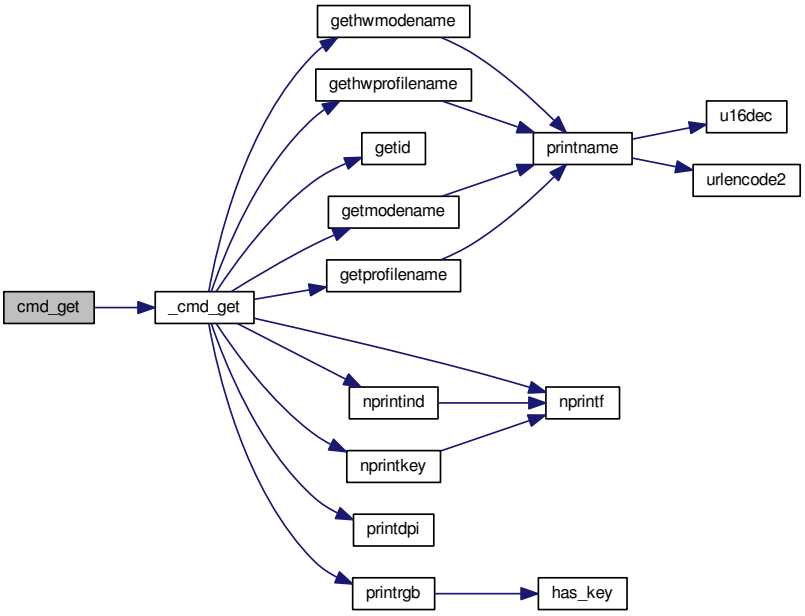
8.33.2.2 void cmd_get (usbdevice * *kb*, usbmode * *mode*, int *nnumber*, int *dummy*, const char * *setting*)

Definition at line 215 of file notify.c.

References _cmd_get(), and imutex.

```
215                                                                                 {
216     pthread_mutex_lock(imutex(kb));
217     _cmd_get(kb, mode, nnumber, setting);
218     pthread_mutex_unlock(imutex(kb));
219 }
```

Here is the call graph for this function:



8.33.2.3 void cmd_notify (usbdevice * *kb*, usbmode * *mode*, int *nnumber*, int *keyindex*, const char * *toggle*)

Definition at line 61 of file notify.c.

References CLEAR_KEYBIT, imutex, N_KEYS_INPUT, usbmode::notify, and SET_KEYBIT.

```
61                                                                                                  {
62     if(keyindex >= N_KEYS_INPUT)
63         return;
64     pthread_mutex_lock(imutex(kb));
65     if(!strcmp(toggle, "on") || *toggle == 0)
66         SET_KEYBIT(mode->notify[nnumber], keyindex);
67     else if(!strcmp(toggle, "off"))
68         CLEAR_KEYBIT(mode->notify[nnumber], keyindex);
69     pthread_mutex_unlock(imutex(kb));
70 }
```

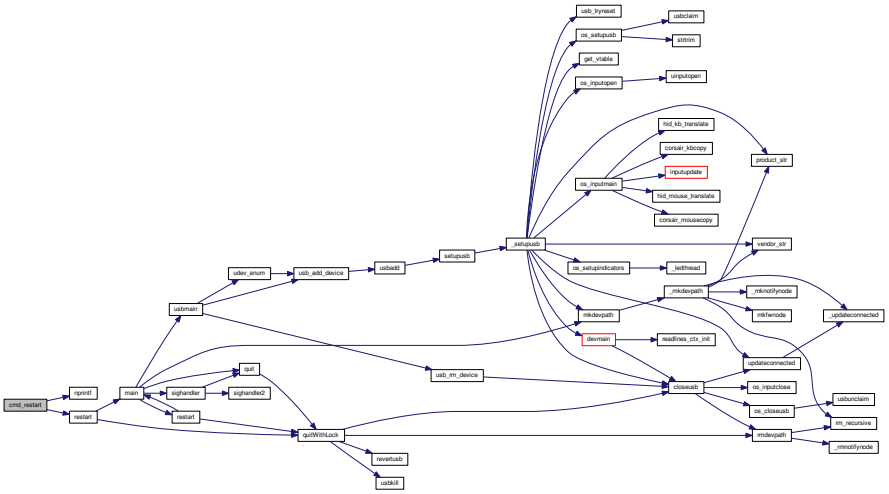
8.33.2.4 void cmd_restart (usbdevice * *kb*, usbmode * *mode*, int *nnumber*, int *dummy*, const char * *content*)

Definition at line 223 of file notify.c.

References ckb_info, nprintf(), and restart().

```
223                                                                                                 {
224     ckb_info("RESTART called with %s\n", content);
225     nprintf(kb, -1, 0, "RESTART called with %s\n", content);
226     restart();
227 }
```

Here is the call graph for this function:



8.33.2.5 void nprintf (usbdevice * *kb*, int *nodenumber*, usbmode * *mode*, const char * *format*, ...)

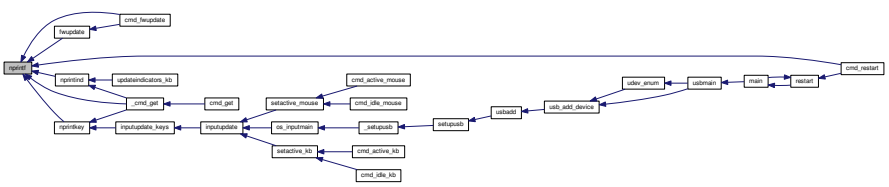
Definition at line 8 of file notify.c.

References INDEX_OF, usbprofile::mode, usbdevice::outfifo, OUTFIFO_MAX, and usbdevice::profile.

Referenced by _cmd_get(), cmd_fwupdate(), cmd_restart(), fwupdate(), nprintind(), and nprintkey().

```
8                                                                                                   {
9       if(!kb)
10          return;
11      usbprofile* profile = kb->profile;
12      va_list va_args;
13      int fifo;
14      if(nodenumber >= 0){
15          // If node number was given, print to that node (if open)
16          if((fifo = kb->outfifo[nodenumber] - 1) != -1){
17              va_start(va_args, format);
18              if(mode)
19                  dprintf(fifo, "mode %d ", INDEX_OF(mode, profile->mode) + 1);
20              vdprintf(fifo, format, va_args);
21          }
22          return;
23      }
24      // Otherwise, print to all nodes
25      for(int i = 0; i < OUTFIFO_MAX; i++){
26          if((fifo = kb->outfifo[i] - 1) != -1){
27              va_start(va_args, format);
28              if(mode)
29                  dprintf(fifo, "mode %d ", INDEX_OF(mode, profile->mode) + 1);
30              vdprintf(fifo, format, va_args);
31          }
32      }
33  }
```

Here is the caller graph for this function:



8.33.2.6 void nprintind (usbdevice * *kb*, int *nnumber*, int *led*, int *on*)

Definition at line 43 of file notify.c.

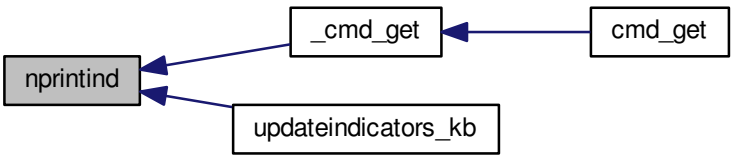
References I_CAPS, I_NUM, I_SCROLL, and nprintf().

Referenced by _cmd_get(), and updateindicators_kb().

```
43                                                                  {
44     const char* name = 0;
45     switch(led){
46     case I_NUM:
47         name = "num";
48         break;
49     case I_CAPS:
50         name = "caps";
51         break;
52     case I_SCROLL:
53         name = "scroll";
54         break;
55     default:
56         return;
57     }
58     nprintf(kb, nnumber, 0, "i %c%s\n", on ? '+' : '-', name);
59 }
```

Here is the call graph for this function:

Here is the caller graph for this function:



8.33.2.7 void nprintkey (usbdevice * *kb*, int *nnumber*, int *keyindex*, int *down*)

Definition at line 35 of file notify.c.

References keymap, key::name, and nprintf().

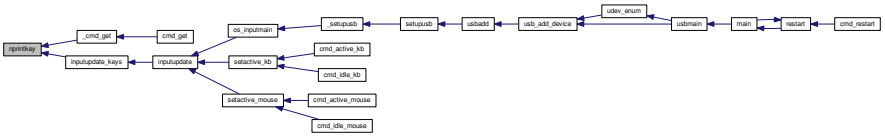
Referenced by _cmd_get(), and inputupdate_keys().

```
35                                                                  {
36     const key* map = keymap + keyindex;
37     if(map->name)
```

```
38         nprintf(kb, nnumber, 0, "key %c%s\n", down ? '+' : '-', map->name);
39     else
40         nprintf(kb, nnumber, 0, "key %c#%d\n", down ? '+' : '-', keyindex);
41 }
```

Here is the call graph for this function:

Here is the caller graph for this function:



8.33.2.8 int restart ( )

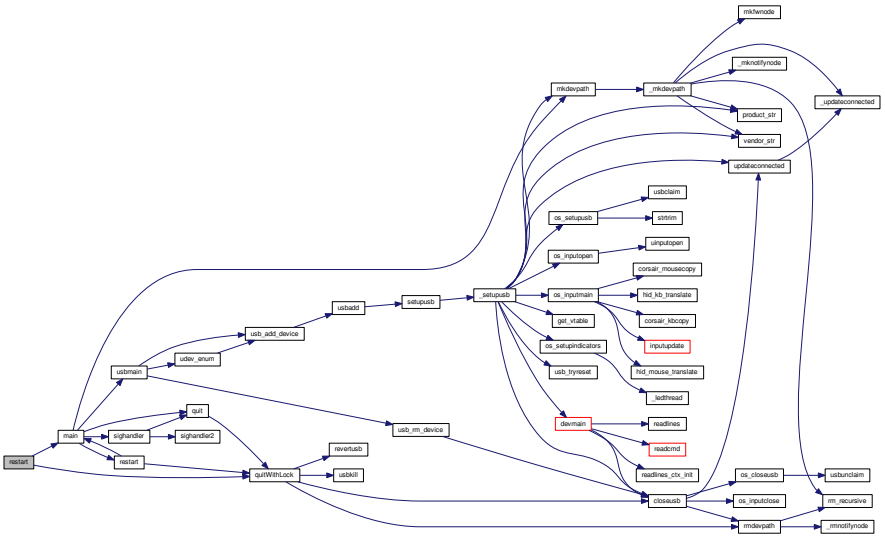
Definition at line 228 of file main.c.

References ckb_err, main(), main_ac, main_av, and quitWithLock().

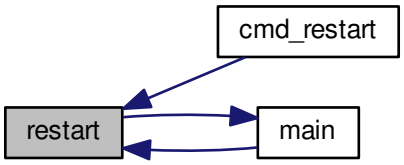
Referenced by cmd_restart(), and main().

```
228             {
229     ckb_err("restart called, running quit without mutex-lock.\n");
230     quitWithLock(0);
231     return main(main_ac, main_av);
232 }
```

Here is the call graph for this function:



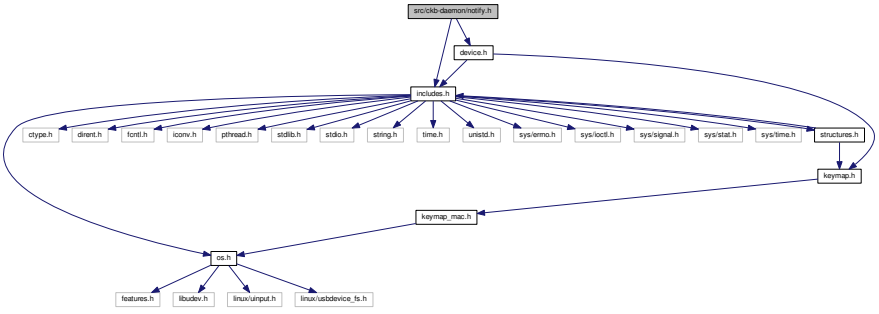
Here is the caller graph for this function:



8.34 src/ckb-daemon/notify.h File Reference

```
#include "includes.h"
#include "device.h"
```

Include dependency graph for notify.h:



This graph shows which files directly or indirectly include this file:

Functions

- void nprintf (usbdevice *kb, int nodenumber, usbmode *mode, const char *format,...)
- void nprintkey (usbdevice *kb, int nnumber, int keyindex, int down)
- void nprintind (usbdevice *kb, int nnumber, int led, int on)
- void cmd_notify (usbdevice *kb, usbmode *mode, int nnumber, int keyindex, const char *toggle)
- void cmd_get (usbdevice *kb, usbmode *mode, int nnumber, int dummy, const char *setting)
- void cmd_restart (usbdevice *kb, usbmode *mode, int nnumber, int dummy, const char *content)

8.34.1 Function Documentation

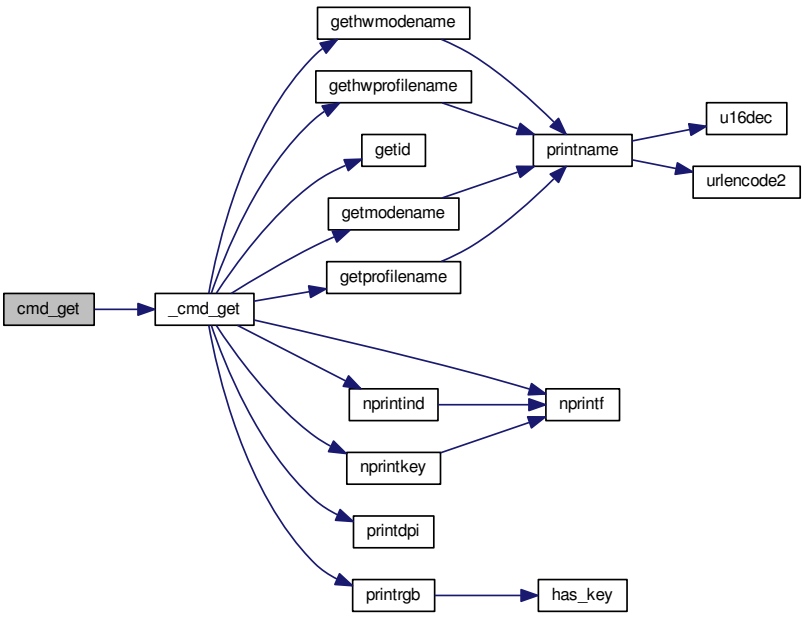
8.34.1.1 void cmd_get (usbdevice * *kb,* usbmode * *mode,* int *nnumber,* int *dummy,* const char * *setting*)

Definition at line 215 of file notify.c.

References _cmd_get(), and imutex.

```
215                                                                                                 {
216     pthread_mutex_lock(imutex(kb));
217     _cmd_get(kb, mode, nnumber, setting);
218     pthread_mutex_unlock(imutex(kb));
219 }
```

Here is the call graph for this function:



8.34.1.2 void cmd_notify (usbdevice * kb, usbmode * mode, int nnumber, int keyindex, const char * toggle)

Definition at line 61 of file notify.c.

References CLEAR_KEYBIT, imutex, N_KEYS_INPUT, usbmode::notify, and SET_KEYBIT.

```
61                                                                                                  {
62     if(keyindex >= N_KEYS_INPUT)
63         return;
64     pthread_mutex_lock(imutex(kb));
65     if(!strcmp(toggle, "on") || *toggle == 0)
66         SET_KEYBIT(mode->notify[nnumber], keyindex);
67     else if(!strcmp(toggle, "off"))
68         CLEAR_KEYBIT(mode->notify[nnumber], keyindex);
69     pthread_mutex_unlock(imutex(kb));
70 }
```

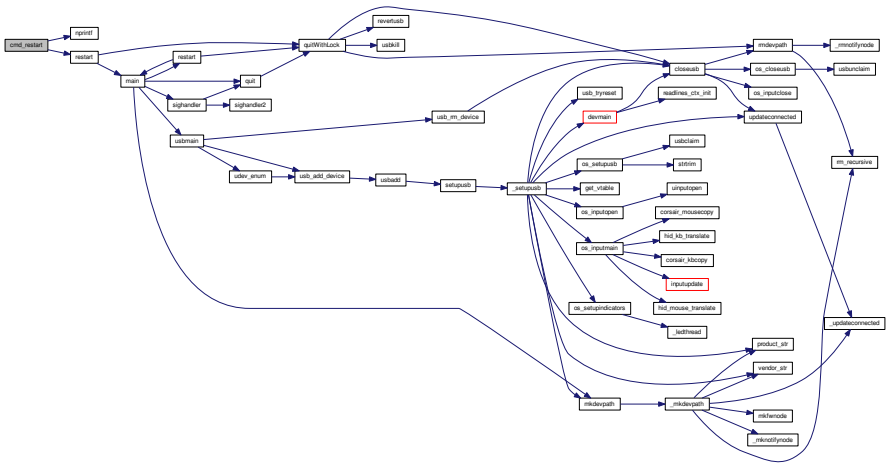
8.34.1.3 void cmd_restart (usbdevice * kb, usbmode * mode, int nnumber, int dummy, const char * content)

Definition at line 223 of file notify.c.

References ckb_info, nprintf(), and restart().

```
223                                                                                                 {
224     ckb_info("RESTART called with %s\n", content);
225     nprintf(kb, -1, 0, "RESTART called with %s\n", content);
226     restart();
227 }
```

Here is the call graph for this function:



8.34.1.4 void nprintf (usbdevice * kb, int nodenumber, usbmode * mode, const char * format, ...)

Definition at line 8 of file notify.c.

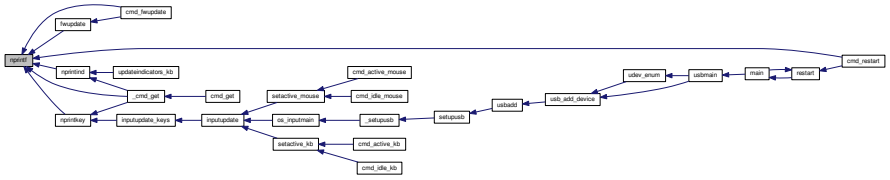
References INDEX_OF, usbprofile::mode, usbdevice::outfifo, OUTFIFO_MAX, and usbdevice::profile.

Referenced by _cmd_get(), cmd_fwupdate(), cmd_restart(), fwupdate(), nprintind(), and nprintkey().

```
8                                                                                                   {
9     if(!kb)
10         return;
11     usbprofile* profile = kb->profile;
12     va_list va_args;
13     int fifo;
```

```
14      if(nodenumber >= 0){
15          // If node number was given, print to that node (if open)
16          if((fifo = kb->outfifo[nodenumber] - 1) != -1){
17              va_start(va_args, format);
18              if(mode)
19                  dprintf(fifo, "mode %d ", INDEX_OF(mode, profile->mode) + 1);
20              vdprintf(fifo, format, va_args);
21          }
22          return;
23      }
24      // Otherwise, print to all nodes
25      for(int i = 0; i < OUTFIFO_MAX; i++){
26          if((fifo = kb->outfifo[i] - 1) != -1){
27              va_start(va_args, format);
28              if(mode)
29                  dprintf(fifo, "mode %d ", INDEX_OF(mode, profile->mode) + 1);
30              vdprintf(fifo, format, va_args);
31          }
32      }
33  }
```

Here is the caller graph for this function:



8.34.1.5 void nprintind (usbdevice * *kb*, int *nnumber*, int *led*, int *on*)

Definition at line 43 of file notify.c.

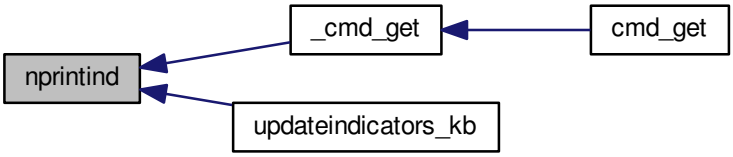
References I_CAPS, I_NUM, I_SCROLL, and nprintf().

Referenced by _cmd_get(), and updateindicators_kb().

```
43                                                          {
44      const char* name = 0;
45      switch(led){
46      case I_NUM:
47          name = "num";
48          break;
49      case I_CAPS:
50          name = "caps";
51          break;
52      case I_SCROLL:
53          name = "scroll";
54          break;
55      default:
56          return;
57      }
58      nprintf(kb, nnumber, 0, "i %c%s\n", on ? '+' : '-', name);
59  }
```

Here is the call graph for this function:

Here is the caller graph for this function:



8.34.1.6 void nprintkey (usbdevice * *kb,* int *nnumber,* int *keyindex,* int *down*)

Definition at line 35 of file notify.c.

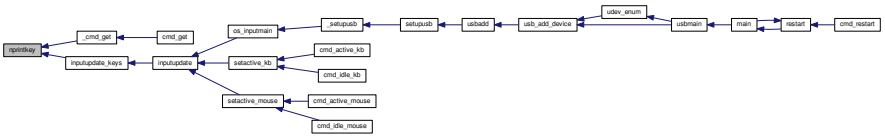
References keymap, key::name, and nprintf().

Referenced by _cmd_get(), and inputupdate_keys().

```
35                                                                  {
36     const key* map = keymap + keyindex;
37     if(map->name)
38         nprintf(kb, nnumber, 0, "key %c%s\n", down ? '+' : '-', map->name);
39     else
40         nprintf(kb, nnumber, 0, "key %c#%d\n", down ? '+' : '-', keyindex);
41 }
```

Here is the call graph for this function:

Here is the caller graph for this function:

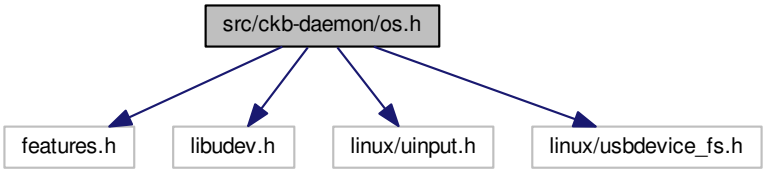


8.35 src/ckb-daemon/os.h File Reference

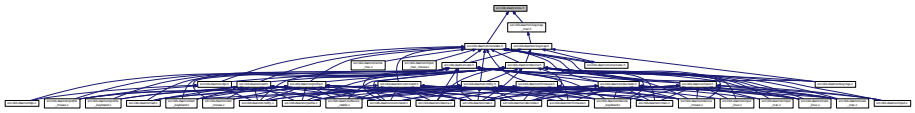
```
#include <features.h>
```

```
#include <libudev.h>
#include <linux/uinput.h>
#include <linux/usbdevice_fs.h>
```

Include dependency graph for os.h:



This graph shows which files directly or indirectly include this file:



Macros

- #define _DEFAULT_SOURCE
- #define _GNU_SOURCE
- #define UINPUT_VERSION 2
- #define euid_guard_start
- #define euid_guard_stop

8.35.1 Macro Definition Documentation

8.35.1.1 #define _DEFAULT_SOURCE

Definition at line 22 of file os.h.

8.35.1.2 #define _GNU_SOURCE

Definition at line 26 of file os.h.

8.35.1.3 #define euid_guard_start

Definition at line 40 of file os.h.

Referenced by mkdevpath(), mknotifynode(), rmdevpath(), rmnotifynode(), and updateconnected().

8.35.1.4 #define euid_guard_stop

Definition at line 41 of file os.h.

Referenced by mkdevpath(), mknotifynode(), rmdevpath(), rmnotifynode(), and updateconnected().

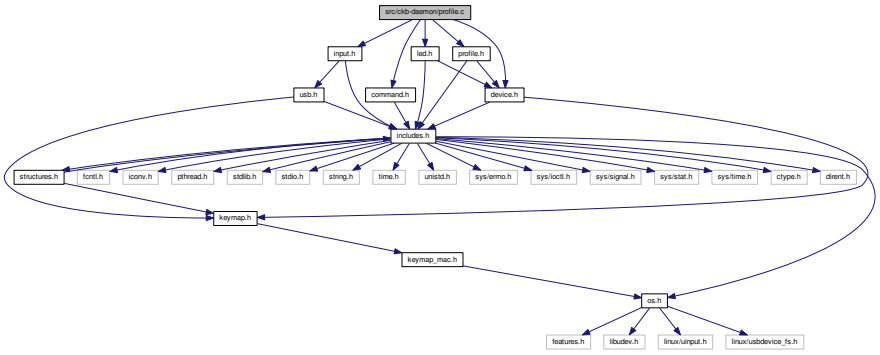
8.35.1.5 #define UINPUT_VERSION 2

Definition at line 35 of file os.h.

8.36 src/ckb-daemon/profile.c File Reference

```
#include "command.h"
#include "device.h"
#include "input.h"
#include "led.h"
#include "profile.h"
```

Include dependency graph for profile.c:



Functions

- void urldecode2 (char *dst, const char *src)
- void urlencode2 (char *dst, const char *src)
- int setid (usbid *id, const char *guid)
- char * getid (usbid *id)
- void u16enc (char *in, ushort *out, size_t *srclen, size_t *dstlen)
- void u16dec (ushort *in, char *out, size_t *srclen, size_t *dstlen)
- void cmd_name (usbdevice *kb, usbmode *mode, int dummy1, int dummy2, const char *name)
- void cmd_profilename (usbdevice *kb, usbmode *dummy1, int dummy2, int dummy3, const char *name)
- char * printname (ushort *name, int length)
- char * getmodename (usbmode *mode)
- char * getprofilename (usbprofile *profile)
- char * gethwmodename (hwprofile *profile, int index)
- char * gethwprofilename (hwprofile *profile)
- void cmd_id (usbdevice *kb, usbmode *mode, int dummy1, int dummy2, const char *id)
- void cmd_profileid (usbdevice *kb, usbmode *mode, int dummy1, int dummy2, const char *id)
- static void initmode (usbmode *mode)
- void allocprofile (usbdevice *kb)
- int loadprofile (usbdevice *kb)
- static void freemode (usbmode *mode)
- void cmd_erase (usbdevice *kb, usbmode *mode, int dummy1, int dummy2, const char *dummy3)
- static void _freeprofile (usbdevice *kb)
- void cmd_eraseprofile (usbdevice *kb, usbmode *dummy1, int dummy2, int dummy3, const char *dummy4)
- void freeprofile (usbdevice *kb)
- void hwtonative (usbprofile *profile, hwprofile *hw, int modecount)
- void nativetohw (usbprofile *profile, hwprofile *hw, int modecount)

Variables

- static iconv_t utf8to16 = 0
- static iconv_t utf16to8 = 0

8.36.1 Function Documentation

8.36.1.1 static void _freeprofile (usbdevice * kb) [static]

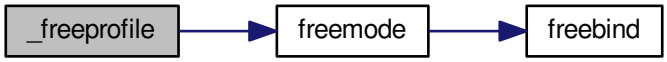
Definition at line 210 of file profile.c.

References freemode(), usbprofile::mode, MODE_COUNT, and usbdevice::profile.

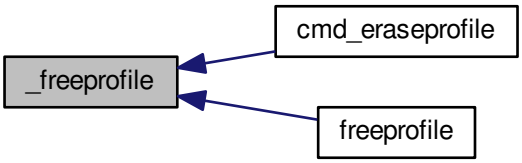
Referenced by cmd_eraseprofile(), and freeprofile().

```
210                                         {
211     usbprofile* profile = kb->profile;
212     if(!profile)
213         return;
214     // Clear all mode data
215     for(int i = 0; i < MODE_COUNT; i++)
216         freemode(profile->mode + i);
217     free(profile);
218     kb->profile = 0;
219 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



8.36.1.2 void allocprofile (usbdevice * kb)

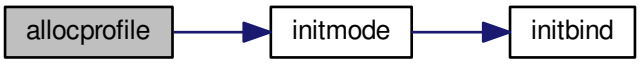
Definition at line 182 of file profile.c.

References usbprofile::currentmode, dpiset::forceupdate, lighting::forceupdate, initmode(), usbprofile::lastdpi, usbprofile::lastlight, usbprofile::mode, MODE_COUNT, and usbdevice::profile.

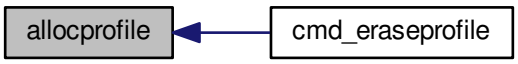
Referenced by cmd_eraseprofile().

```
182                                 {
183     if(kb->profile)
184         return;
185     usbprofile* profile = kb->profile = calloc(1, sizeof(
      usbprofile));
186     for(int i = 0; i < MODE_COUNT; i++)
187         initmode(profile->mode + i);
188     profile->currentmode = profile->mode;
189     profile->lastlight.forceupdate = profile->lastdpi.
      forceupdate = 1;
190 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



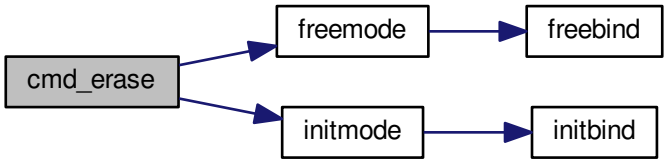
8.36.1.3 void cmd_erase (usbdevice * *kb*, usbmode * *mode*, int *dummy1*, int *dummy2*, const char * *dummy3*)

Definition at line 203 of file profile.c.

References freemode(), imutex, and initmode().

```
203                                                                                 {
204     pthread_mutex_lock(imutex(kb));
205     freemode(mode);
206     initmode(mode);
207     pthread_mutex_unlock(imutex(kb));
208 }
```

Here is the call graph for this function:



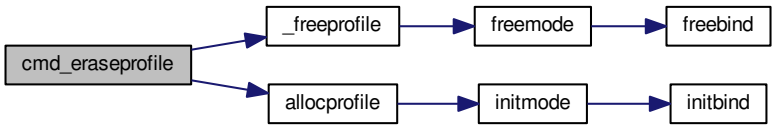
8.36.1.4 void cmd_eraseprofile (usbdevice * *kb*, usbmode * *dummy1*, int *dummy2*, int *dummy3*, const char * *dummy4*)

Definition at line 221 of file profile.c.

References _freeprofile(), allocprofile(), and imutex.

```
221                                                                                                 {
222     pthread_mutex_lock(imutex(kb));
223     _freeprofile(kb);
224     allocprofile(kb);
225     pthread_mutex_unlock(imutex(kb));
226 }
```

Here is the call graph for this function:



8.36.1.5 void cmd_id (usbdevice * *kb*, usbmode * *mode*, int *dummy1*, int *dummy2*, const char * *id*)

Definition at line 160 of file profile.c.

References usbmode::id, usbid::modified, and setid().

```
160                                                                                                 {
161     // ID is either a GUID or an 8-digit hex number
162     int newmodified;
163     if(!setid(&mode->id, id) && sscanf(id, "%08x", &newmodified) == 1)
164         memcpy(mode->id.modified, &newmodified, sizeof(newmodified));
165 }
```

Here is the call graph for this function:

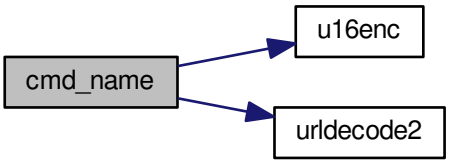
8.36.1.6 void cmd_name (usbdevice * *kb*, usbmode * *mode*, int *dummy1*, int *dummy2*, const char * *name*)

Definition at line 117 of file profile.c.

References MD_NAME_LEN, usbmode::name, u16enc(), and urldecode2().

```
117                                                                                                 {
118     char decoded[strlen(name) + 1];
119     urldecode2(decoded, name);
120     size_t srclen = strlen(decoded), dstlen = MD_NAME_LEN;
121     u16enc(decoded, mode->name, &srclen, &dstlen);
122 }
```

Here is the call graph for this function:



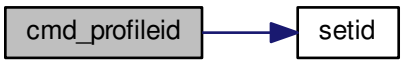
8.36.1.7 void cmd_profileid (usbdevice * *kb*, usbmode * *mode*, int *dummy1*, int *dummy2*, const char * *id*)

Definition at line 167 of file profile.c.

References usbprofile::id, usbid::modified, usbdevice::profile, and setid().

```
167                                                                                                 {
168     usbprofile* profile = kb->profile;
169     int newmodified;
170     if(!setid(&profile->id, id) && sscanf(id, "%08x", &newmodified) == 1)
171         memcpy(profile->id.modified, &newmodified, sizeof(newmodified));
172
173 }
```

Here is the call graph for this function:



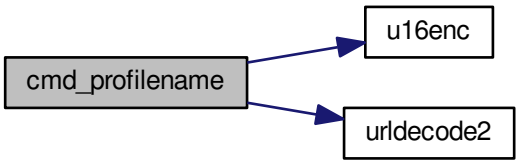
8.36.1.8 void cmd_profilename (usbdevice * kb, usbmode * dummy1, int dummy2, int dummy3, const char * name)

Definition at line 124 of file profile.c.

References usbprofile::name, PR_NAME_LEN, usbdevice::profile, u16enc(), and urldecode2().

```
124                                                                                                 {
125     usbprofile* profile = kb->profile;
126     char decoded[strlen(name) + 1];
127     urldecode2(decoded, name);
128     size_t srclen = strlen(decoded), dstlen = PR_NAME_LEN;
129     u16enc(decoded, profile->name, &srclen, &dstlen);
130 }
```

Here is the call graph for this function:



8.36.1.9 static void freemode (usbmode * mode) [static]

Definition at line 198 of file profile.c.

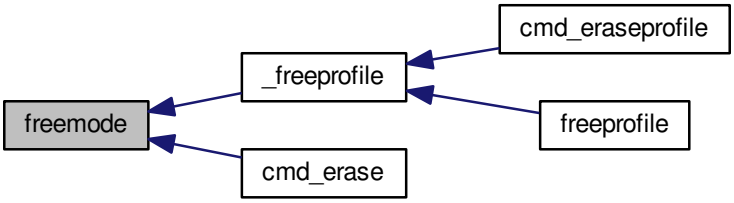
References usbmode::bind, and freebind().

Referenced by _freeprofile(), and cmd_erase().

```
198                                     {
199     freebind(&mode->bind);
200     memset(mode, 0, sizeof(*mode));
201 }
```

Here is the call graph for this function:

Here is the caller graph for this function:



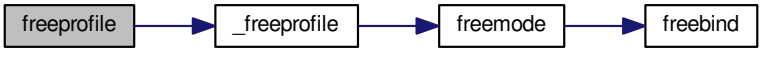
8.36.1.10 void freeprofile (usbdevice * *kb*)

Definition at line 228 of file profile.c.

References _freeprofile(), and usbdevice::hw.

```
228                                {
229     _freeprofile(kb);
230     // Also free HW profile
231     free(kb->hw);
232     kb->hw = 0;
233 }
```

Here is the call graph for this function:



8.36.1.11 char* gethwmodename (hwprofile * *profile,* int *index*)

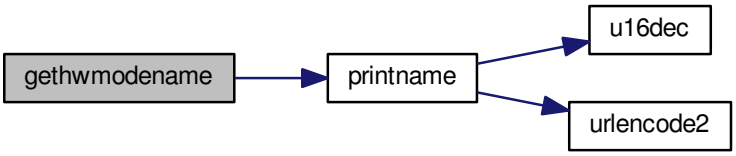
Definition at line 152 of file profile.c.

References MD_NAME_LEN, hwprofile::name, and printname().

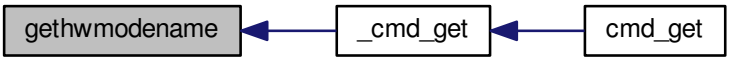
Referenced by _cmd_get().

```
152                                                 {
153     return printname(profile->name[index + 1], MD_NAME_LEN);
154 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



8.36.1.12 char* gethwprofilename (hwprofile * *profile*)

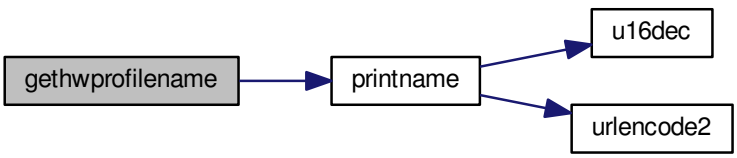
Definition at line 156 of file profile.c.

References MD_NAME_LEN, hwprofile::name, and printname().

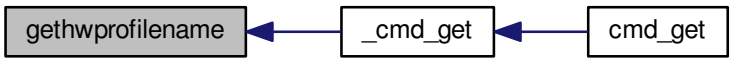
Referenced by _cmd_get().

```
156                                                 {
157     return printname(profile->name[0], MD_NAME_LEN);
158 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



8.36.1.13 char* getid (usbid * *id*)

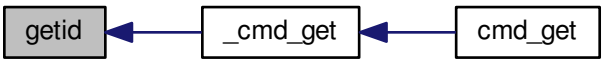
Definition at line 79 of file profile.c.

References usbid::guid.

Referenced by _cmd_get().

```
79                      {
80     int32_t data1;
81     int16_t data2, data3, data4a;
82     char data4b[6];
83     memcpy(&data1, id->guid + 0x0, 4);
84     memcpy(&data2, id->guid + 0x4, 2);
85     memcpy(&data3, id->guid + 0x6, 2);
86     memcpy(&data4a, id->guid + 0x8, 2);
87     memcpy(data4b, id->guid + 0xA, 6);
88     char* guid = malloc(39);
89     snprintf(guid, 39, "{%08X-%04hX-%04hX-%04hX-%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX}",
90              data1, data2, data3, data4a, data4b[0], data4b[1], data4b[2], data4b[3], data4b[4], data4b[5])
     ;
91     return guid;
92 }
```

Here is the caller graph for this function:



8.36.1.14 char* getmodename (usbmode * *mode*)

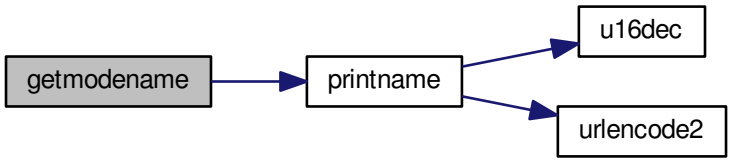
Definition at line 144 of file profile.c.

References MD_NAME_LEN, usbmode::name, and printname().

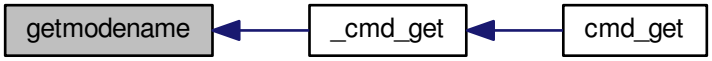
Referenced by _cmd_get().

```
144                                 {
145     return printname(mode->name, MD_NAME_LEN);
146 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



8.36.1.15 char* getprofilename (usbprofile * *profile*)

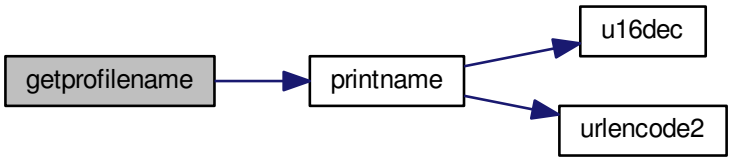
Definition at line 148 of file profile.c.

References usbprofile::name, PR_NAME_LEN, and printname().

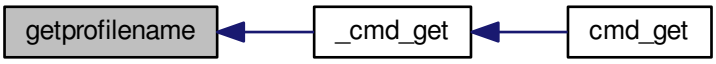
Referenced by _cmd_get().

```
148                                          {
149     return printname(profile->name, PR_NAME_LEN);
150 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



8.36.1.16 void hwtonative (usbprofile * *profile,* hwprofile * *hw,* int *modecount*)

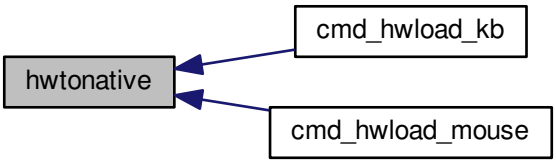
Definition at line 235 of file profile.c.

References usbmode::dpi, hwprofile::dpi, dpiset::forceupdate, lighting::forceupdate, usbmode::id, usbprofile::id, hwprofile::id, usbprofile::lastdpi, usbprofile::lastlight, usbmode::light, hwprofile::light, MD_NAME_LEN, usbprofile::mode, usbmode::name, usbprofile::name, hwprofile::name, and PR_NAME_LEN.

Referenced by cmd_hwload_kb(), and cmd_hwload_mouse().

```
235                                                                    {
236     // Copy the profile name and ID
237     memcpy(profile->name, hw->name[0], PR_NAME_LEN * 2);
238     memcpy(&profile->id, hw->id, sizeof(usbid));
239     // Copy the mode settings
240     for(int i = 0; i < modecount; i++){
241         usbmode* mode = profile->mode + i;
242         memcpy(mode->name, hw->name[i + 1], MD_NAME_LEN * 2);
243         memcpy(&mode->id, hw->id + i + 1, sizeof(usbid));
244         memcpy(&mode->light, hw->light + i, sizeof(lighting));
245         memcpy(&mode->dpi, hw->dpi + i, sizeof(dpiset));
246         // Set a force update on the light/DPI since they've been overwritten
247         mode->light.forceupdate = mode->dpi.forceupdate = 1;
248     }
249     profile->lastlight.forceupdate = profile->lastdpi.
      forceupdate = 1;
250 }
```

Here is the caller graph for this function:



8.36.1.17 static void initmode (usbmode * *mode*) [static]

Definition at line 175 of file profile.c.

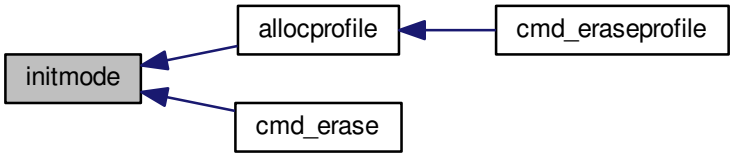
References usbmode::bind, usbmode::dpi, dpiset::forceupdate, lighting::forceupdate, initbind(), and usbmode::light.

Referenced by allocprofile(), and cmd_erase().

```
175                                     {
176     memset(mode, 0, sizeof(*mode));
177     mode->light.forceupdate = 1;
178     mode->dpi.forceupdate = 1;
179     initbind(&mode->bind);
180 }
```

Here is the call graph for this function:

Here is the caller graph for this function:



8.36.1.18 int loadprofile (usbdevice * *kb*)

Definition at line 192 of file profile.c.

References hwloadprofile.

```
192                                 {
193     if(hwloadprofile(kb, 1))
194         return -1;
195     return 0;
196 }
```

8.36.1.19 void nativetohw (usbprofile * *profile,* hwprofile * *hw,* int *modecount*)

Definition at line 252 of file profile.c.

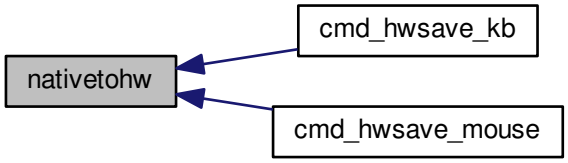
References usbmode::dpi, hwprofile::dpi, usbmode::id, usbprofile::id, hwprofile::id, usbmode::light, hwprofile::light, MD_NAME_LEN, usbprofile::mode, usbmode::name, usbprofile::name, hwprofile::name, and PR_NAME_LEN.

Referenced by cmd_hwsave_kb(), and cmd_hwsave_mouse().

```
252                                                                 {
253     // Copy name and ID
254     memcpy(hw->name[0], profile->name, PR_NAME_LEN * 2);
255     memcpy(hw->id, &profile->id, sizeof(usbid));
256     // Copy the mode settings
```

```
257     for(int i = 0; i < modecount; i++){
258         usbmode* mode = profile->mode + i;
259         memcpy(hw->name[i + 1], mode->name, MD_NAME_LEN * 2);
260         memcpy(hw->id + i + 1, &mode->id, sizeof(usbid));
261         memcpy(hw->light + i, &mode->light, sizeof(lighting));
262         memcpy(hw->dpi + i, &mode->dpi, sizeof(dpiset));
263     }
264 }
```

Here is the caller graph for this function:



8.36.1.20 char* printname (ushort * *name*, int *length*)

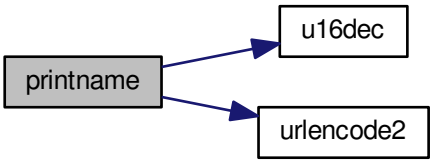
Definition at line 132 of file profile.c.

References u16dec(), and urlencode2().

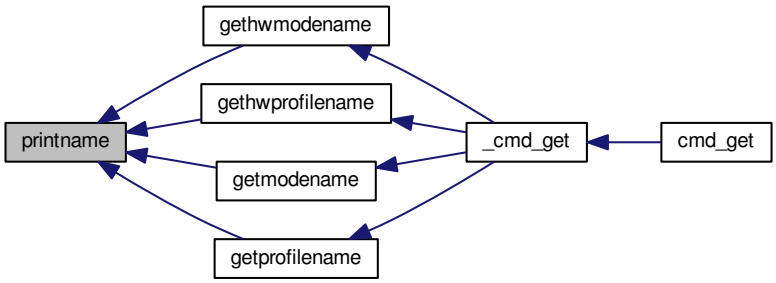
Referenced by gethwmodename(), gethwprofilename(), getmodename(), and getprofilename().

```
132                                         {
133     // Convert the name to UTF-8
134     char* buffer = calloc(1, length * 4 - 3);
135     size_t srclen = length, dstlen = length * 4 - 4;
136     u16dec(name, buffer, &srclen, &dstlen);
137     // URL-encode it
138     char* buffer2 = malloc(strlen(buffer) * 3 + 1);
139     urlencode2(buffer2, buffer);
140     free(buffer);
141     return buffer2;
142 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



8.36.1.21 int setid (usbid * *id*, const char * *guid*)

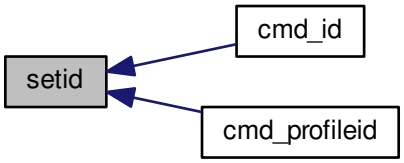
Definition at line 64 of file profile.c.

References usbid::guid.

Referenced by cmd_id(), and cmd_profileid().

```
64                                      {
65     int32_t data1;
66     int16_t data2, data3, data4a;
67     char data4b[6];
68     if(sscanf(guid, "{%08X-%04hX-%04hX-%04hX-%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX}",
69               &data1, &data2, &data3, &data4a, data4b, data4b + 1, data4b + 2, data4b + 3, data4b + 4,
      data4b + 5) != 10)
70         return 0;
71     memcpy(id->guid + 0x0, &data1, 4);
72     memcpy(id->guid + 0x4, &data2, 2);
73     memcpy(id->guid + 0x6, &data3, 2);
74     memcpy(id->guid + 0x8, &data4a, 2);
75     memcpy(id->guid + 0xA, data4b, 6);
76     return 1;
77 }
```

Here is the caller graph for this function:



8.36.1.22 void u16dec (ushort * *in*, char * *out*, size_t * *srclen*, size_t * *dstlen*)

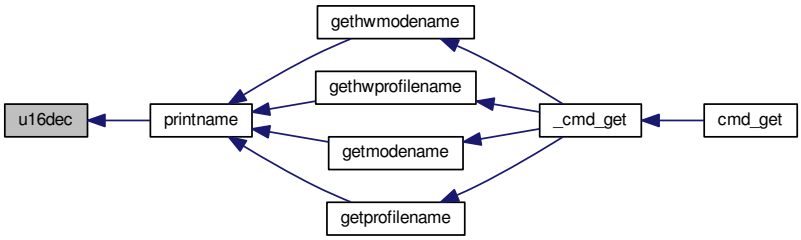
Definition at line 105 of file profile.c.

References utf16to8.

Referenced by printname().

```
105                                                                     {
106     if(!utf16to8)
107         utf16to8 = iconv_open("UTF-8", "UTF-16LE");
108     size_t srclen2 = 0, srclenmax = *srclen;
109     for(; srclen2 < srclenmax; srclen2++){
110         if(!in[srclen2])
111             break;
112     }
113     *srclen = srclen2 * 2;
114     iconv(utf16to8, (char**)&in, srclen, &out, dstlen);
115 }
```

Here is the caller graph for this function:



8.36.1.23 void u16enc (char * *in*, ushort * *out*, size_t * *srclen*, size_t * *dstlen*)

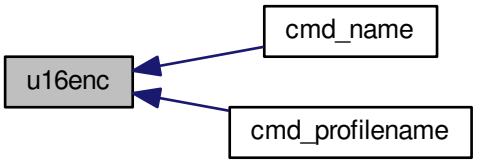
Definition at line 97 of file profile.c.

References utf8to16.

Referenced by cmd_name(), and cmd_profilename().

```
97                                                                      {
98     if(!utf8to16)
99         utf8to16 = iconv_open("UTF-16LE", "UTF-8");
100     memset(out, 0, *dstlen * 2);
101     *dstlen = *dstlen * 2 - 2;
102     iconv(utf8to16, &in, srclen, (char**)&out, dstlen);
103 }
```

Here is the caller graph for this function:



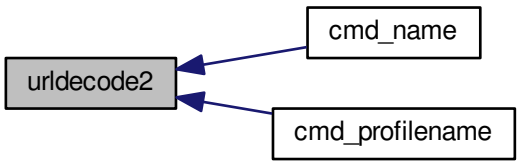
8.36.1.24 void urldecode2 (char * *dst*, const char * *src*)

Definition at line 8 of file profile.c.

Referenced by cmd_name(), and cmd_profilename().

```
8                                               {
9     char a, b;
10     char s;
11     while((s = *src)){
12         if((s == '%') &&
13                 ((a = src[1]) && (b = src[2])) &&
14                 (isxdigit(a) && isxdigit(b))){
15             if(a >= 'a')
16                 a -= 'a'-'A';
17             if(a >= 'A')
18                 a -= 'A' - 10;
19             else
20                 a -= '0';
21             if(b >= 'a')
22                 b -= 'a'-'A';
23             if(b >= 'A')
24                 b -= 'A' - 10;
25             else
26                 b -= '0';
27             *dst++ = 16 * a + b;
28             src += 3;
29         } else {
30             *dst++ = s;
31             src++;
32         }
33     }
34     *dst = '\0';
35 }
```

Here is the caller graph for this function:



8.36.1.25 void urlencode2 (char * *dst*, const char * *src*)

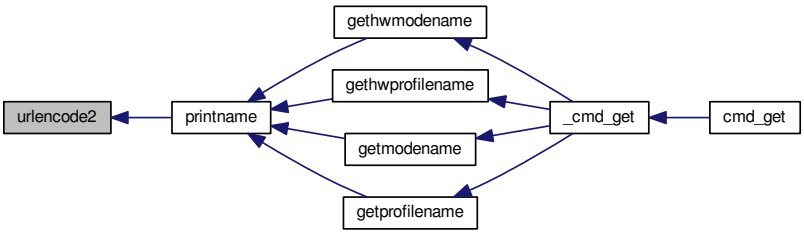
Definition at line 37 of file profile.c.

Referenced by printname().

```
37                                               {
38     char s;
39     while((s = *src++)){
40         if(s <= ',' || s == '/' ||
41                 (s >= ':' && s <= '@') ||
42                 s == '[' || s == ']' ||
43                 s >= 0x7F){
44             char a = s >> 4, b = s & 0xF;
45             if(a >= 10)
46                 a += 'A' - 10;
47             else
48                 a += '0';
49             if(b >= 10)
50                 b += 'A' - 10;
51             else
```

```
52                 b += '0';
53             dst[0] = '%';
54             dst[1] = a;
55             dst[2] = b;
56             dst += 3;
57         } else
58             *dst++ = s;
59     }
60     *dst = '\0';
61 }
```

Here is the caller graph for this function:



8.36.2 Variable Documentation

8.36.2.1 iconv_t utf16to8 = 0 [static]

Definition at line 95 of file profile.c.

Referenced by u16dec().

8.36.2.2 iconv_t utf8to16 = 0 [static]

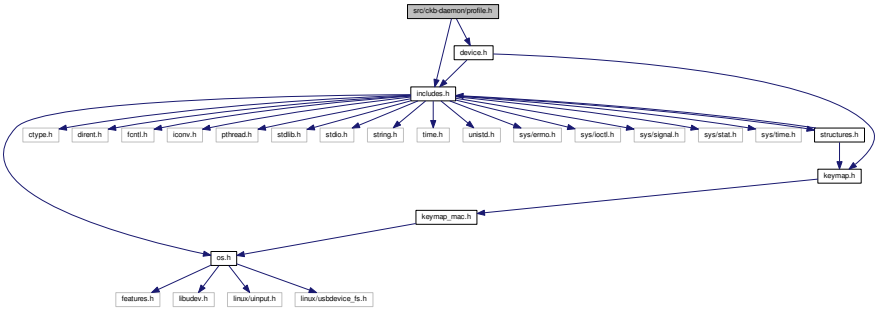
Definition at line 95 of file profile.c.

Referenced by u16enc().

8.37 src/ckb-daemon/profile.h File Reference

```
#include "includes.h"
#include "device.h"
```

Include dependency graph for profile.h:



This graph shows which files directly or indirectly include this file:

Macros

- #define hwloadprofile(kb, apply) (kb)->vtable->hwload(kb, 0, 0, apply, 0)

Functions

- void allocprofile (usbdevice *kb)
- int loadprofile (usbdevice *kb)
- void freeprofile (usbdevice *kb)
- void cmd_erase (usbdevice *kb, usbmode *mode, int dummy1, int dummy2, const char *dummy3)
- void cmd_eraseprofile (usbdevice *kb, usbmode *dummy1, int dummy2, int dummy3, const char *dummy4)
- void cmd_name (usbdevice *kb, usbmode *mode, int dummy1, int dummy2, const char *name)
- void cmd_profilename (usbdevice *kb, usbmode *dummy1, int dummy2, int dummy3, const char *name)
- char * getmodename (usbmode *mode)
- char * getprofilename (usbprofile *profile)
- char * gethwmodename (hwprofile *profile, int index)
- char * gethwprofilename (hwprofile *profile)
- int setid (usbid *id, const char *guid)
- char * getid (usbid *id)
- void hwtonative (usbprofile *profile, hwprofile *hw, int modecount)
- void nativetohw (usbprofile *profile, hwprofile *hw, int modecount)
- void cmd_id (usbdevice *kb, usbmode *mode, int dummy1, int dummy2, const char *id)
- void cmd_profileid (usbdevice *kb, usbmode *mode, int dummy1, int dummy2, const char *id)
- int cmd_hwload_kb (usbdevice *kb, usbmode *dummy1, int dummy2, int apply, const char *dummy3)
- int cmd_hwload_mouse (usbdevice *kb, usbmode *dummy1, int dummy2, int apply, const char *dummy3)
- int cmd_hwsave_kb (usbdevice *kb, usbmode *dummy1, int dummy2, int dummy3, const char *dummy4)
- int cmd_hwsave_mouse (usbdevice *kb, usbmode *dummy1, int dummy2, int dummy3, const char *dummy4)

8.37.1 Macro Definition Documentation

8.37.1.1 #define hwloadprofile(*kb, apply*) (kb)->vtable->hwload(kb, 0, 0, apply, 0)

Definition at line 52 of file profile.h.

Referenced by _start_dev(), and loadprofile().

8.37.2 Function Documentation

8.37.2.1 void allocprofile (usbdevice * *kb*)

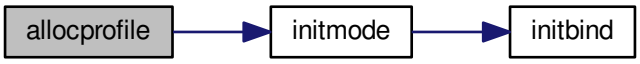
Definition at line 182 of file profile.c.

References usbprofile::currentmode, dpiset::forceupdate, lighting::forceupdate, initmode(), usbprofile::lastdpi, usbprofile::lastlight, usbprofile::mode, MODE_COUNT, and usbdevice::profile.

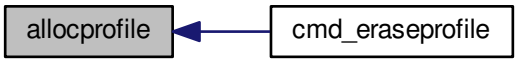
Referenced by cmd_eraseprofile().

```
182                                 {
183     if(kb->profile)
184         return;
185     usbprofile* profile = kb->profile = calloc(1, sizeof(
    usbprofile));
186     for(int i = 0; i < MODE_COUNT; i++)
187         initmode(profile->mode + i);
188     profile->currentmode = profile->mode;
189     profile->lastlight.forceupdate = profile->lastdpi.
    forceupdate = 1;
190 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



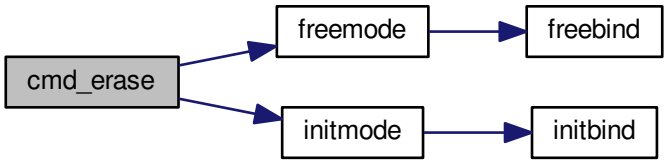
8.37.2.2 void cmd_erase (usbdevice * *kb*, usbmode * *mode*, int *dummy1*, int *dummy2*, const char * *dummy3*)

Definition at line 203 of file profile.c.

References freemode(), imutex, and initmode().

```
203                                                                 {
204     pthread_mutex_lock(imutex(kb));
205     freemode(mode);
206     initmode(mode);
207     pthread_mutex_unlock(imutex(kb));
208 }
```

Here is the call graph for this function:



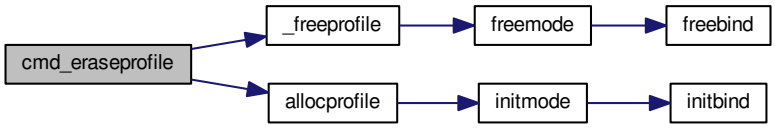
8.37.2.3 void cmd_eraseprofile (usbdevice * *kb*, usbmode * *dummy1*, int *dummy2*, int *dummy3*, const char * *dummy4*)

Definition at line 221 of file profile.c.

References _freeprofile(), allocprofile(), and imutex.

```
221                                                                                                 {
222     pthread_mutex_lock(imutex(kb));
223     _freeprofile(kb);
224     allocprofile(kb);
225     pthread_mutex_unlock(imutex(kb));
226 }
```

Here is the call graph for this function:



8.37.2.4 int cmd_hwload_kb (usbdevice * *kb*, usbmode * *dummy1*, int *dummy2*, int *apply*, const char * *dummy3*)

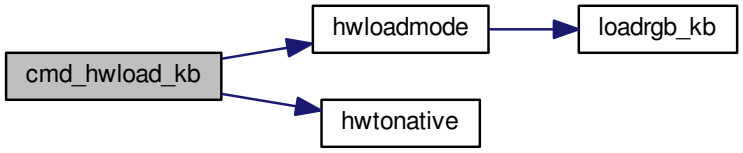
Definition at line 16 of file profile_keyboard.c.

References DELAY_LONG, usbdevice::hw, hwloadmode(), HWMODE_K70, HWMODE_K95, hwtonative(), hwprofile::id, IS_K95, MSG_SIZE, hwprofile::name, PR_NAME_LEN, usbdevice::profile, and usbrecv.

```
16                                                                                                  {
17     DELAY_LONG(kb);
18     hwprofile* hw = calloc(1, sizeof(hwprofile));
19     // Ask for profile and mode IDs
20     uchar data_pkt[2][MSG_SIZE] = {
21         { 0x0e, 0x15, 0x01, 0 },
22         { 0x0e, 0x16, 0x01, 0 }
23     };
24     uchar in_pkt[MSG_SIZE];
25     int modes = (IS_K95(kb) ? HWMODE_K95 : HWMODE_K70);
26     for(int i = 0; i <= modes; i++){
27         data_pkt[0][3] = i;
28         if(!usbrecv(kb, data_pkt[0], in_pkt)){
```

```
29              free(hw);
30              return -1;
31          }
32          memcpy(hw->id + i, in_pkt + 4, sizeof(usbid));
33      }
34      // Ask for profile name
35      if(!usbrecv(kb, data_pkt[1], in_pkt)){
36          free(hw);
37          return -1;
38      }
39      memcpy(hw->name[0], in_pkt + 4, PR_NAME_LEN * 2);
40      // Load modes
41      for(int i = 0; i < modes; i++){
42          if(hwloadmode(kb, hw, i)){
43              free(hw);
44              return -1;
45          }
46      }
47      // Make the profile active (if requested)
48      if(apply)
49          hwtonative(kb->profile, hw, modes);
50      // Free the existing profile (if any)
51      free(kb->hw);
52      kb->hw = hw;
53      DELAY_LONG(kb);
54      return 0;
55  }
```

Here is the call graph for this function:



8.37.2.5 int cmd_hwload_mouse (usbdevice * *kb*, usbmode * *dummy1*, int *dummy2*, int *apply*, const char * *dummy3*)

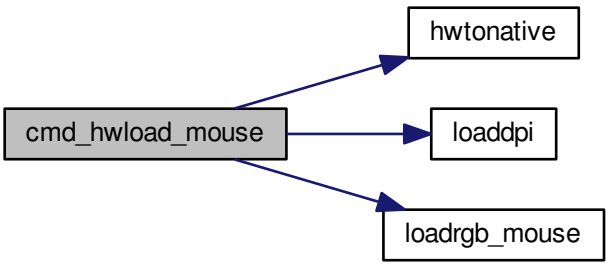
Definition at line 6 of file profile_mouse.c.

References DELAY_LONG, hwprofile::dpi, usbdevice::hw, hwtonative(), hwprofile::id, hwprofile::light, loaddpi(), loadrgb_mouse(), MSG_SIZE, hwprofile::name, PR_NAME_LEN, usbdevice::profile, and usbrecv.

```
6                                                                                   {
7       DELAY_LONG(kb);
8       hwprofile* hw = calloc(1, sizeof(hwprofile));
9       // Ask for profile and mode IDs
10      uchar data_pkt[2][MSG_SIZE] = {
11          { 0x0e, 0x15, 0x01, 0 },
12          { 0x0e, 0x16, 0x01, 0 }
13      };
14      uchar in_pkt[MSG_SIZE];
15      for(int i = 0; i <= 1; i++){
16          data_pkt[0][3] = i;
17          if(!usbrecv(kb, data_pkt[0], in_pkt)){
18              free(hw);
19              return -1;
20          }
21          memcpy(hw->id + i, in_pkt + 4, sizeof(usbid));
22      }
23      // Ask for profile and mode names
24      for(int i = 0; i <= 1; i++){
25          data_pkt[1][3] = i;
26          if(!usbrecv(kb, data_pkt[1],in_pkt)){
27              free(hw);
28              return -1;
29          }
```

```
30          memcpy(hw->name[i], in_pkt + 4, PR_NAME_LEN * 2);
31      }
32
33      // Load the RGB and DPI settings
34      if(loadrgb_mouse(kb, hw->light, 0)
35              || loaddpi(kb, hw->dpi, hw->light)){
36          free(hw);
37          return -1;
38      }
39
40      // Make the profile active (if requested)
41      if(apply)
42          hwtonative(kb->profile, hw, 1);
43      // Free the existing profile (if any)
44      free(kb->hw);
45      kb->hw = hw;
46      DELAY_LONG(kb);
47      return 0;
48  }
```

Here is the call graph for this function:



8.37.2.6 int cmd_hwsave_kb (usbdevice * *kb*, usbmode * *dummy1*, int *dummy2*, int *dummy3*, const char * *dummy4*)

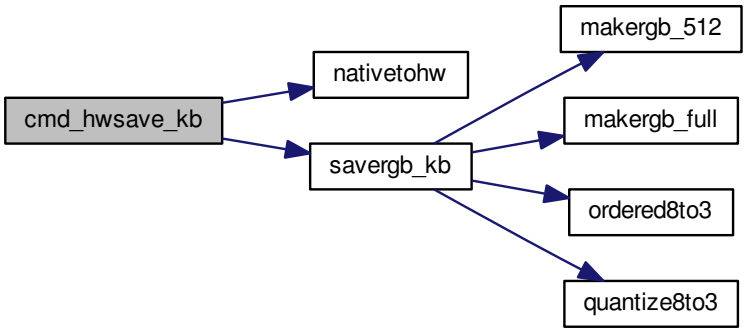
Definition at line 57 of file profile_keyboard.c.

References DELAY_LONG, usbdevice::hw, HWMODE_K70, HWMODE_K95, hwprofile::id, IS_K95, hwprofile::light, MD_NAME_LEN, MSG_SIZE, hwprofile::name, nativetohw(), usbdevice::profile, savergb_kb(), and usbsend.

```
57                                                                                  {
58      DELAY_LONG(kb);
59      hwprofile* hw = kb->hw;
60      if(!hw)
61          hw = kb->hw = calloc(1, sizeof(hwprofile));
62      int modes = (IS_K95(kb) ? HWMODE_K95 : HWMODE_K70);
63      nativetohw(kb->profile, hw, modes);
64      // Save the profile and mode names
65      uchar data_pkt[2][MSG_SIZE] = {
66          { 0x07, 0x16, 0x01, 0 },
67          { 0x07, 0x15, 0x01, 0 },
68      };
69      // Save the mode names
70      for(int i = 0; i <= modes; i++){
71          data_pkt[0][3] = i;
72          memcpy(data_pkt[0] + 4, hw->name[i], MD_NAME_LEN * 2);
73          if(!usbsend(kb, data_pkt[0], 1))
74              return -1;
75      }
76      // Save the IDs
77      for(int i = 0; i <= modes; i++){
78          data_pkt[1][3] = i;
79          memcpy(data_pkt[1] + 4, hw->id + i, sizeof(usbid));
80          if(!usbsend(kb, data_pkt[1], 1))
81              return -1;
82      }
```

```
83      // Save the RGB data
84      for(int i = 0; i < modes; i++){
85          if(savergb_kb(kb, hw->light + i, i))
86              return -1;
87      }
88      DELAY_LONG(kb);
89      return 0;
90  }
```

Here is the call graph for this function:



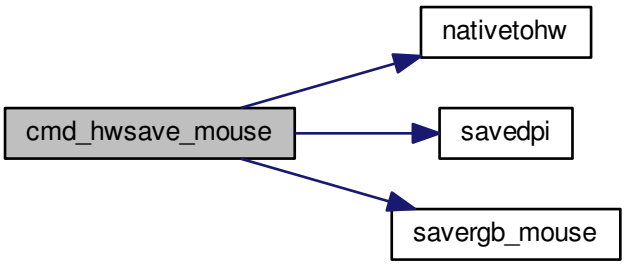
8.37.2.7 int cmd_hwsave_mouse (usbdevice * kb, usbmode * dummy1, int dummy2, int dummy3, const char * dummy4)

Definition at line 50 of file profile_mouse.c.

References DELAY_LONG, hwprofile::dpi, usbdevice::hw, hwprofile::id, hwprofile::light, MD_NAME_LEN, MSG_SIZE, hwprofile::name, nativetohw(), usbdevice::profile, savedpi(), savergb_mouse(), and usbsend.

```
50                                                                                                  {
51      DELAY_LONG(kb);
52      hwprofile* hw = kb->hw;
53      if(!hw)
54          hw = kb->hw = calloc(1, sizeof(hwprofile));
55      nativetohw(kb->profile, hw, 1);
56      // Save the profile and mode names
57      uchar data_pkt[2][MSG_SIZE] = {
58          { 0x07, 0x16, 0x01, 0 },
59          { 0x07, 0x15, 0x01, 0 },
60      };
61      for(int i = 0; i <= 1; i++){
62          data_pkt[0][3] = i;
63          memcpy(data_pkt[0] + 4, hw->name[i], MD_NAME_LEN * 2);
64          if(!usbsend(kb, data_pkt[0], 1))
65              return -1;
66      }
67      // Save the IDs
68      for(int i = 0; i <= 1; i++){
69          data_pkt[1][3] = i;
70          memcpy(data_pkt[1] + 4, hw->id + i, sizeof(usbid));
71          if(!usbsend(kb, data_pkt[1], 1))
72              return -1;
73      }
74      // Save the RGB data for the non-DPI zones
75      if(savergb_mouse(kb, hw->light, 0))
76          return -1;
77      // Save the DPI data (also saves RGB for those states)
78      if(savedpi(kb, hw->dpi, hw->light))
79          return -1;
80      DELAY_LONG(kb);
81      return 0;
82  }
```

Here is the call graph for this function:



8.37.2.8 void cmd_id (usbdevice * *kb*, usbmode * *mode*, int *dummy1*, int *dummy2*, const char * *id*)

Definition at line 160 of file profile.c.

References usbmode::id, usbid::modified, and setid().

```
160                                                                                 {
161     // ID is either a GUID or an 8-digit hex number
162     int newmodified;
163     if(!setid(&mode->id, id) && sscanf(id, "%08x", &newmodified) == 1)
164         memcpy(mode->id.modified, &newmodified, sizeof(newmodified));
165 }
```

Here is the call graph for this function:

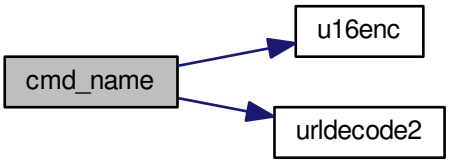
8.37.2.9 void cmd_name (usbdevice * *kb*, usbmode * *mode*, int *dummy1*, int *dummy2*, const char * *name*)

Definition at line 117 of file profile.c.

References MD_NAME_LEN, usbmode::name, u16enc(), and urldecode2().

```
117                                                                                 {
118     char decoded[strlen(name) + 1];
119     urldecode2(decoded, name);
120     size_t srclen = strlen(decoded), dstlen = MD_NAME_LEN;
121     u16enc(decoded, mode->name, &srclen, &dstlen);
122 }
```

Here is the call graph for this function:



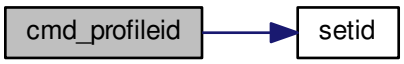
8.37.2.10 void cmd_profileid (usbdevice * kb, usbmode * mode, int dummy1, int dummy2, const char * id)

Definition at line 167 of file profile.c.

References usbprofile::id, usbid::modified, usbdevice::profile, and setid().

```
167                                                                                 {
168     usbprofile* profile = kb->profile;
169     int newmodified;
170     if(!setid(&profile->id, id) && sscanf(id, "%08x", &newmodified) == 1)
171         memcpy(profile->id.modified, &newmodified, sizeof(newmodified));
172
173 }
```

Here is the call graph for this function:



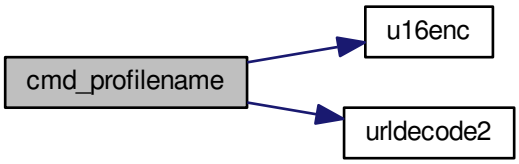
8.37.2.11 void cmd_profilename (usbdevice * kb, usbmode * dummy1, int dummy2, int dummy3, const char * name)

Definition at line 124 of file profile.c.

References usbprofile::name, PR_NAME_LEN, usbdevice::profile, u16enc(), and urldecode2().

```
124                                                                                 {
125     usbprofile* profile = kb->profile;
126     char decoded[strlen(name) + 1];
127     urldecode2(decoded, name);
128     size_t srclen = strlen(decoded), dstlen = PR_NAME_LEN;
129     u16enc(decoded, profile->name, &srclen, &dstlen);
130 }
```

Here is the call graph for this function:



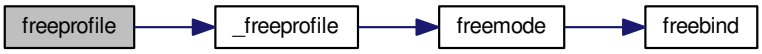
8.37.2.12 void freeprofile (usbdevice * *kb*)

Definition at line 228 of file profile.c.

References _freeprofile(), and usbdevice::hw.

```
228                                {
229     _freeprofile(kb);
230     // Also free HW profile
231     free(kb->hw);
232     kb->hw = 0;
233 }
```

Here is the call graph for this function:



8.37.2.13 char* gethwmodename (hwprofile * *profile,* int *index*)

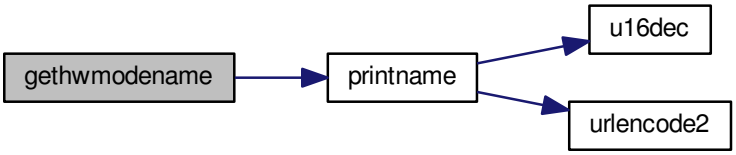
Definition at line 152 of file profile.c.

References MD_NAME_LEN, hwprofile::name, and printname().

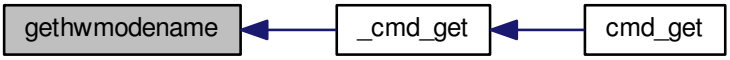
Referenced by _cmd_get().

```
152                                                {
153     return printname(profile->name[index + 1], MD_NAME_LEN);
154 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



8.37.2.14 char* gethwprofilename (hwprofile * *profile*)

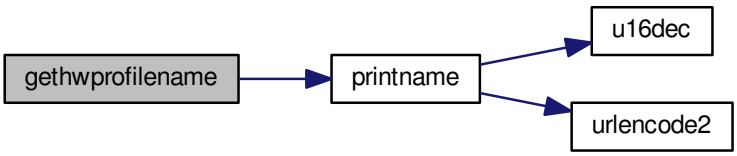
Definition at line 156 of file profile.c.

References MD_NAME_LEN, hwprofile::name, and printname().

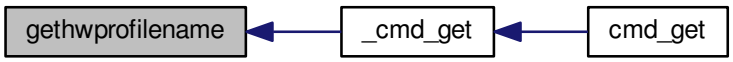
Referenced by _cmd_get().

```
156                                          {
157     return printname(profile->name[0], MD_NAME_LEN);
158 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



8.37.2.15 char* getid (usbid * *id*)

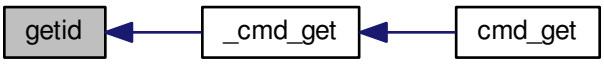
Definition at line 79 of file profile.c.

References usbid::guid.

Referenced by _cmd_get().

```
79                      {
80     int32_t data1;
81     int16_t data2, data3, data4a;
82     char data4b[6];
83     memcpy(&data1, id->guid + 0x0, 4);
84     memcpy(&data2, id->guid + 0x4, 2);
85     memcpy(&data3, id->guid + 0x6, 2);
86     memcpy(&data4a, id->guid + 0x8, 2);
87     memcpy(data4b, id->guid + 0xA, 6);
88     char* guid = malloc(39);
89     snprintf(guid, 39, "{%08X-%04hX-%04hX-%04hX-%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX}",
90              data1, data2, data3, data4a, data4b[0], data4b[1], data4b[2], data4b[3], data4b[4], data4b[5])
     ;
91     return guid;
92 }
```

Here is the caller graph for this function:



8.37.2.16 char* getmodename (usbmode * *mode*)

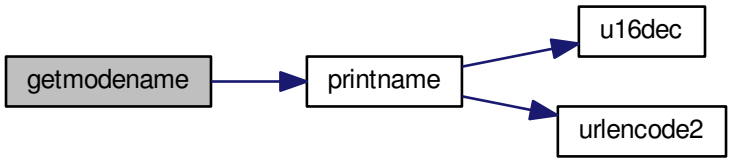
Definition at line 144 of file profile.c.

References MD_NAME_LEN, usbmode::name, and printname().

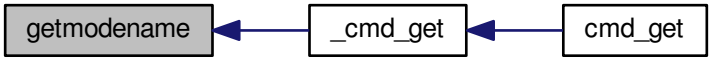
Referenced by _cmd_get().

```
144                                 {
145     return printname(mode->name, MD_NAME_LEN);
146 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



8.37.2.17 char* getprofilename (usbprofile * *profile*)

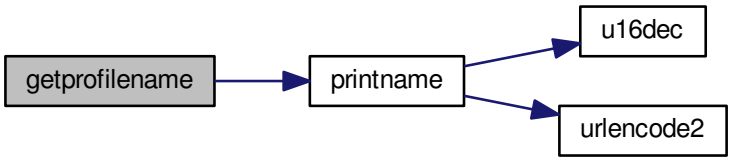
Definition at line 148 of file profile.c.

References usbprofile::name, PR_NAME_LEN, and printname().

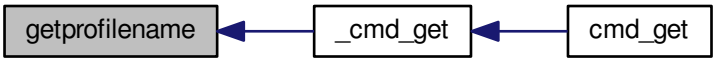
Referenced by _cmd_get().

```
148                                          {
149     return printname(profile->name, PR_NAME_LEN);
150 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



8.37.2.18 void hwtonative (usbprofile * *profile,* hwprofile * *hw,* int *modecount*)

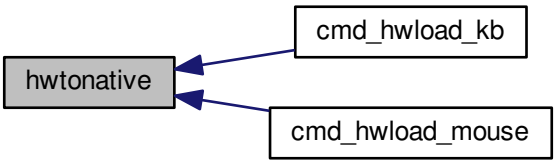
Definition at line 235 of file profile.c.

References usbmode::dpi, hwprofile::dpi, dpiset::forceupdate, lighting::forceupdate, usbmode::id, usbprofile::id, hwprofile::id, usbprofile::lastdpi, usbprofile::lastlight, usbmode::light, hwprofile::light, MD_NAME_LEN, usbprofile::mode, usbmode::name, usbprofile::name, hwprofile::name, and PR_NAME_LEN.

Referenced by cmd_hwload_kb(), and cmd_hwload_mouse().

```
235                                                                    {
236     // Copy the profile name and ID
237     memcpy(profile->name, hw->name[0], PR_NAME_LEN * 2);
238     memcpy(&profile->id, hw->id, sizeof(usbid));
239     // Copy the mode settings
240     for(int i = 0; i < modecount; i++){
241         usbmode* mode = profile->mode + i;
242         memcpy(mode->name, hw->name[i + 1], MD_NAME_LEN * 2);
243         memcpy(&mode->id, hw->id + i + 1, sizeof(usbid));
244         memcpy(&mode->light, hw->light + i, sizeof(lighting));
245         memcpy(&mode->dpi, hw->dpi + i, sizeof(dpiset));
246         // Set a force update on the light/DPI since they've been overwritten
247         mode->light.forceupdate = mode->dpi.forceupdate = 1;
248     }
249     profile->lastlight.forceupdate = profile->lastdpi.
      forceupdate = 1;
250 }
```

Here is the caller graph for this function:



8.37.2.19 int loadprofile (usbdevice * *kb*)

Definition at line 192 of file profile.c.

References hwloadprofile.

```
192                                 {
193     if(hwloadprofile(kb, 1))
194         return -1;
195     return 0;
196 }
```

8.37.2.20 void nativetohw (usbprofile * *profile,* hwprofile * *hw,* int *modecount*)

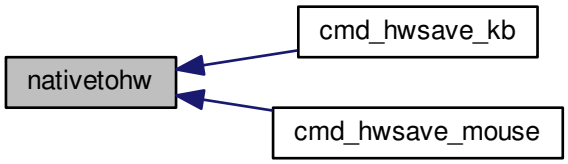
Definition at line 252 of file profile.c.

References usbmode::dpi, hwprofile::dpi, usbmode::id, usbprofile::id, hwprofile::id, usbmode::light, hwprofile::light, MD_NAME_LEN, usbprofile::mode, usbmode::name, usbprofile::name, hwprofile::name, and PR_NAME_LEN.

Referenced by cmd_hwsave_kb(), and cmd_hwsave_mouse().

```
252                                                                 {
253     // Copy name and ID
254     memcpy(hw->name[0], profile->name, PR_NAME_LEN * 2);
255     memcpy(hw->id, &profile->id, sizeof(usbid));
256     // Copy the mode settings
257     for(int i = 0; i < modecount; i++){
258         usbmode* mode = profile->mode + i;
259         memcpy(hw->name[i + 1], mode->name, MD_NAME_LEN * 2);
260         memcpy(hw->id + i + 1, &mode->id, sizeof(usbid));
261         memcpy(hw->light + i, &mode->light, sizeof(lighting));
262         memcpy(hw->dpi + i, &mode->dpi, sizeof(dpiset));
263     }
264 }
```

Here is the caller graph for this function:



8.37.2.21 int setid (usbid * *id,* const char * *guid*)

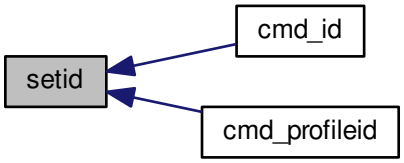
Definition at line 64 of file profile.c.

References usbid::guid.

Referenced by cmd_id(), and cmd_profileid().

```
64                                      {
65     int32_t data1;
66     int16_t data2, data3, data4a;
67     char data4b[6];
68     if(sscanf(guid, "{%08X-%04hX-%04hX-%04hX-%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX}",
69               &data1, &data2, &data3, &data4a, data4b, data4b + 1, data4b + 2, data4b + 3, data4b + 4,
      data4b + 5) != 10)
70         return 0;
71     memcpy(id->guid + 0x0, &data1, 4);
72     memcpy(id->guid + 0x4, &data2, 2);
73     memcpy(id->guid + 0x6, &data3, 2);
74     memcpy(id->guid + 0x8, &data4a, 2);
75     memcpy(id->guid + 0xA, data4b, 6);
76     return 1;
77 }
```

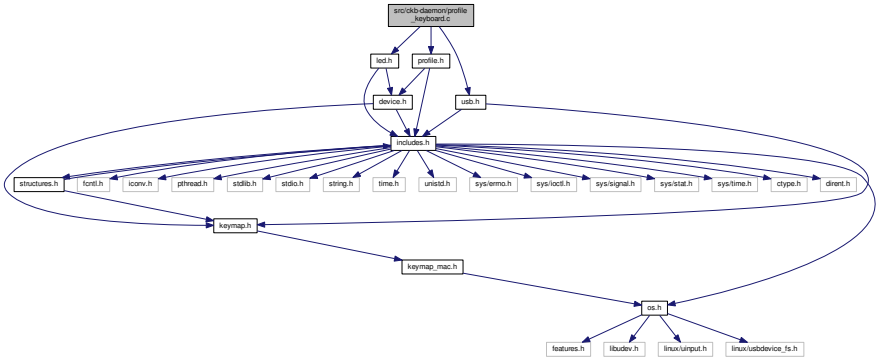
Here is the caller graph for this function:



8.38 src/ckb-daemon/profile_keyboard.c File Reference

```
#include "profile.h"
#include "usb.h"
#include "led.h"
```

Include dependency graph for profile_keyboard.c:



Functions

- static int hwloadmode (usbdevice ∗kb, hwprofile ∗hw, int mode)
- int cmd_hwload_kb (usbdevice ∗kb, usbmode ∗dummy1, int dummy2, int apply, const char ∗dummy3)
- int cmd_hwsave_kb (usbdevice ∗kb, usbmode ∗dummy1, int dummy2, int dummy3, const char ∗dummy4)

8.38.1 Function Documentation

8.38.1.1 int cmd_hwload_kb (usbdevice ∗ *kb,* usbmode ∗ *dummy1,* int *dummy2,* int *apply,* const char ∗ *dummy3*)

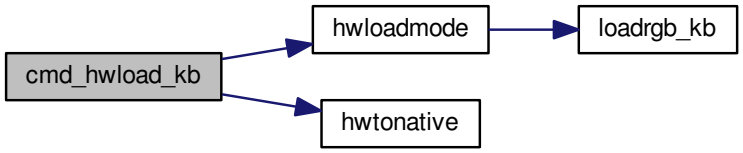
Definition at line 16 of file profile_keyboard.c.

References DELAY_LONG, usbdevice::hw, hwloadmode(), HWMODE_K70, HWMODE_K95, hwtonative(), hwprofile::id, IS_K95, MSG_SIZE, hwprofile::name, PR_NAME_LEN, usbdevice::profile, and usbrecv.

```
16                                                                              {
17     DELAY_LONG(kb);
18     hwprofile* hw = calloc(1, sizeof(hwprofile));
19     // Ask for profile and mode IDs
```

```
20      uchar data_pkt[2][MSG_SIZE] = {
21          { 0x0e, 0x15, 0x01, 0 },
22          { 0x0e, 0x16, 0x01, 0 }
23      };
24      uchar in_pkt[MSG_SIZE];
25      int modes = (IS_K95(kb) ? HWMODE_K95 : HWMODE_K70);
26      for(int i = 0; i <= modes; i++){
27          data_pkt[0][3] = i;
28          if(!usbrecv(kb, data_pkt[0], in_pkt)){
29              free(hw);
30              return -1;
31          }
32          memcpy(hw->id + i, in_pkt + 4, sizeof(usbid));
33      }
34      // Ask for profile name
35      if(!usbrecv(kb, data_pkt[1], in_pkt)){
36          free(hw);
37          return -1;
38      }
39      memcpy(hw->name[0], in_pkt + 4, PR_NAME_LEN * 2);
40      // Load modes
41      for(int i = 0; i < modes; i++){
42          if(hwloadmode(kb, hw, i)){
43              free(hw);
44              return -1;
45          }
46      }
47      // Make the profile active (if requested)
48      if(apply)
49          hwtonative(kb->profile, hw, modes);
50      // Free the existing profile (if any)
51      free(kb->hw);
52      kb->hw = hw;
53      DELAY_LONG(kb);
54      return 0;
55  }
```

Here is the call graph for this function:



8.38.1.2 int cmd_hwsave_kb (usbdevice * *kb*, usbmode * *dummy1*, int *dummy2*, int *dummy3*, const char * *dummy4*)

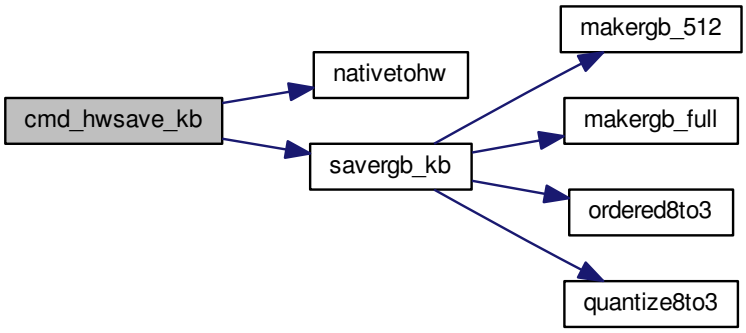
Definition at line 57 of file profile_keyboard.c.

References DELAY_LONG, usbdevice::hw, HWMODE_K70, HWMODE_K95, hwprofile::id, IS_K95, hwprofile::light, MD_NAME_LEN, MSG_SIZE, hwprofile::name, nativetohw(), usbdevice::profile, savergb_kb(), and usbsend.

```
57                                                                                  {
58      DELAY_LONG(kb);
59      hwprofile* hw = kb->hw;
60      if(!hw)
61          hw = kb->hw = calloc(1, sizeof(hwprofile));
62      int modes = (IS_K95(kb) ? HWMODE_K95 : HWMODE_K70);
63      nativetohw(kb->profile, hw, modes);
64      // Save the profile and mode names
65      uchar data_pkt[2][MSG_SIZE] = {
66          { 0x07, 0x16, 0x01, 0 },
67          { 0x07, 0x15, 0x01, 0 },
68      };
69      // Save the mode names
70      for(int i = 0; i <= modes; i++){
71          data_pkt[0][3] = i;
```

```
72          memcpy(data_pkt[0] + 4, hw->name[i], MD_NAME_LEN * 2);
73          if(!usbsend(kb, data_pkt[0], 1))
74              return -1;
75      }
76      // Save the IDs
77      for(int i = 0; i <= modes; i++){
78          data_pkt[1][3] = i;
79          memcpy(data_pkt[1] + 4, hw->id + i, sizeof(usbid));
80          if(!usbsend(kb, data_pkt[1], 1))
81              return -1;
82      }
83      // Save the RGB data
84      for(int i = 0; i < modes; i++){
85          if(savergb_kb(kb, hw->light + i, i))
86              return -1;
87      }
88      DELAY_LONG(kb);
89      return 0;
90  }
```

Here is the call graph for this function:



8.38.1.3 static int hwloadmode (usbdevice * *kb*, hwprofile * *hw*, int *mode*) [static]

Definition at line 5 of file profile_keyboard.c.

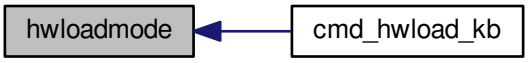
References hwprofile::light, loadrgb_kb(), MD_NAME_LEN, MSG_SIZE, hwprofile::name, and usbrecv.

Referenced by cmd_hwload_kb().

```
5                                                       {
6       // Ask for mode's name
7       uchar data_pkt[MSG_SIZE] = { 0x0e, 0x16, 0x01, mode + 1, 0 };
8       uchar in_pkt[MSG_SIZE];
9       if(!usbrecv(kb, data_pkt, in_pkt))
10          return -1;
11       memcpy(hw->name[mode + 1], in_pkt + 4, MD_NAME_LEN * 2);
12      // Load the RGB setting
13      return loadrgb_kb(kb, hw->light + mode, mode);
14  }
```

Here is the call graph for this function:

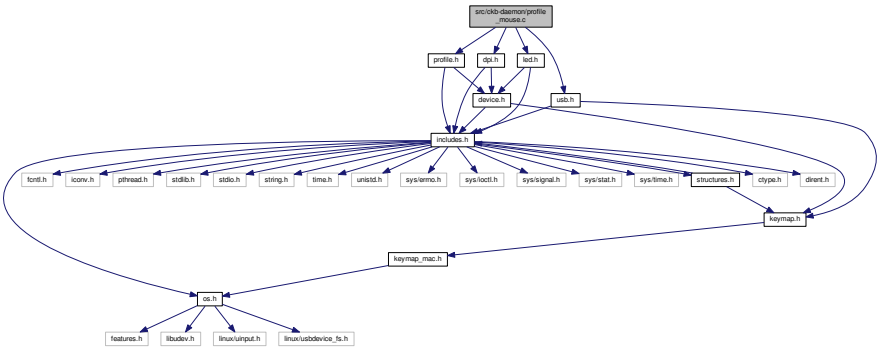
Here is the caller graph for this function:



8.39 src/ckb-daemon/profile_mouse.c File Reference

```
#include "dpi.h"
#include "profile.h"
#include "usb.h"
#include "led.h"
```

Include dependency graph for profile_mouse.c:



Functions

- int cmd_hwload_mouse (usbdevice ∗kb, usbmode ∗dummy1, int dummy2, int apply, const char ∗dummy3)
- int cmd_hwsave_mouse (usbdevice ∗kb, usbmode ∗dummy1, int dummy2, int dummy3, const char ∗dummy4)

8.39.1 Function Documentation

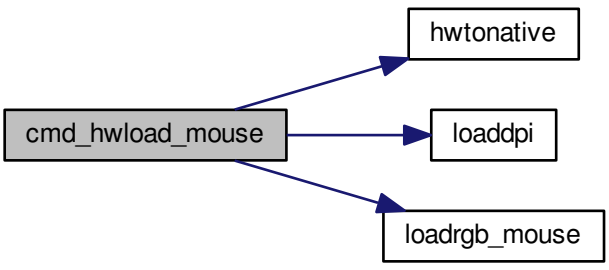
8.39.1.1 int cmd_hwload_mouse (usbdevice * *kb*, usbmode * *dummy1*, int *dummy2*, int *apply*, const char * *dummy3*)

Definition at line 6 of file profile_mouse.c.

References DELAY_LONG, hwprofile::dpi, usbdevice::hw, hwtonative(), hwprofile::id, hwprofile::light, loaddpi(), loadrgb_mouse(), MSG_SIZE, hwprofile::name, PR_NAME_LEN, usbdevice::profile, and usbrecv.

```
6                                                                                                   {
7       DELAY_LONG(kb);
8       hwprofile* hw = calloc(1, sizeof(hwprofile));
9       // Ask for profile and mode IDs
10       uchar data_pkt[2][MSG_SIZE] = {
11           { 0x0e, 0x15, 0x01, 0 },
12           { 0x0e, 0x16, 0x01, 0 }
13       };
14       uchar in_pkt[MSG_SIZE];
15       for(int i = 0; i <= 1; i++){
16           data_pkt[0][3] = i;
17           if(!usbrecv(kb, data_pkt[0], in_pkt)){
18               free(hw);
19               return -1;
20           }
21           memcpy(hw->id + i, in_pkt + 4, sizeof(usbid));
22       }
23       // Ask for profile and mode names
24       for(int i = 0; i <= 1; i++){
25           data_pkt[1][3] = i;
26           if(!usbrecv(kb, data_pkt[1],in_pkt)){
27               free(hw);
28               return -1;
29           }
30           memcpy(hw->name[i], in_pkt + 4, PR_NAME_LEN * 2);
31       }
32
33       // Load the RGB and DPI settings
34       if(loadrgb_mouse(kb, hw->light, 0)
35               || loaddpi(kb, hw->dpi, hw->light)){
36           free(hw);
37           return -1;
38       }
39
40       // Make the profile active (if requested)
41       if(apply)
42           hwtonative(kb->profile, hw, 1);
43       // Free the existing profile (if any)
44       free(kb->hw);
45       kb->hw = hw;
46       DELAY_LONG(kb);
47       return 0;
48   }
```

Here is the call graph for this function:



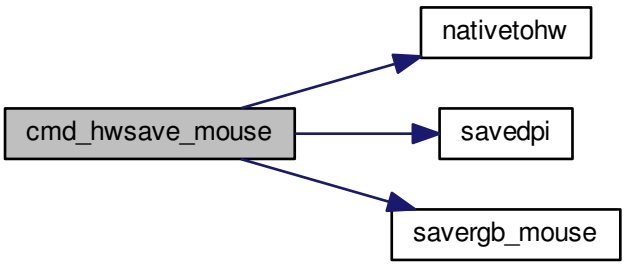
8.39.1.2 int cmd_hwsave_mouse (usbdevice * *kb*, usbmode * *dummy1*, int *dummy2*, int *dummy3*, const char * *dummy4*)

Definition at line 50 of file profile_mouse.c.

References DELAY_LONG, hwprofile::dpi, usbdevice::hw, hwprofile::id, hwprofile::light, MD_NAME_LEN, MSG_SIZE, hwprofile::name, nativetohw(), usbdevice::profile, savedpi(), savergb_mouse(), and usbsend.

```
50                                                                                                  {
51     DELAY_LONG(kb);
52     hwprofile* hw = kb->hw;
53     if(!hw)
54         hw = kb->hw = calloc(1, sizeof(hwprofile));
55     nativetohw(kb->profile, hw, 1);
56     // Save the profile and mode names
57     uchar data_pkt[2][MSG_SIZE] = {
58         { 0x07, 0x16, 0x01, 0 },
59         { 0x07, 0x15, 0x01, 0 },
60     };
61     for(int i = 0; i <= 1; i++){
62         data_pkt[0][3] = i;
63         memcpy(data_pkt[0] + 4, hw->name[i], MD_NAME_LEN * 2);
64         if(!usbsend(kb, data_pkt[0], 1))
65             return -1;
66     }
67     // Save the IDs
68     for(int i = 0; i <= 1; i++){
69         data_pkt[1][3] = i;
70         memcpy(data_pkt[1] + 4, hw->id + i, sizeof(usbid));
71         if(!usbsend(kb, data_pkt[1], 1))
72             return -1;
73     }
74     // Save the RGB data for the non-DPI zones
75     if(savergb_mouse(kb, hw->light, 0))
76         return -1;
77     // Save the DPI data (also saves RGB for those states)
78     if(savedpi(kb, hw->dpi, hw->light))
79         return -1;
80     DELAY_LONG(kb);
81     return 0;
82 }
```

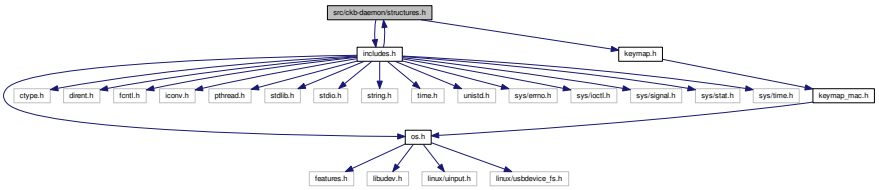
Here is the call graph for this function:



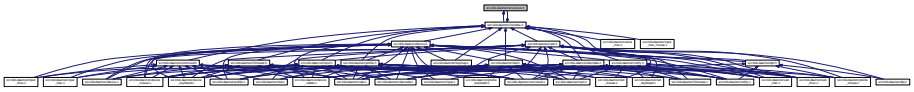
8.40 src/ckb-daemon/structures.h File Reference

```
#include "includes.h"
#include "keymap.h"
```

Include dependency graph for structures.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct usbid
- struct macroaction
- struct keymacro
- struct binding
- struct dpiset
- struct lighting
- struct usbmode
- struct usbprofile
- struct hwprofile
- struct usbinput
- struct usbdevice

Macros

- #define SET_KEYBIT(array, index) do { (array)[(index) / 8] |= 1 << ((index) % 8); } while(0)
- #define CLEAR_KEYBIT(array, index) do { (array)[(index) / 8] &= ~(1 << ((index) % 8)); } while(0)
- #define I_NUM 1
- #define I_CAPS 2
- #define I_SCROLL 4
- #define OUTFIFO_MAX 10
- #define MACRO_MAX 1024
- #define DPI_COUNT 6
- #define LIFT_MIN 1
- #define LIFT_MAX 5
- #define MD_NAME_LEN 16
- #define PR_NAME_LEN 16
- #define MODE_COUNT 6
- #define HWMODE_K70 1
- #define HWMODE_K95 3
- #define HWMODE_MAX 3
- #define FEAT_RGB 0x001
- #define FEAT_MONOCHROME 0x002
- #define FEAT_POLLRATE 0x004

- #define FEAT_ADJRATE 0x008
- #define FEAT_BIND 0x010
- #define FEAT_NOTIFY 0x020
- #define FEAT_FWVERSION 0x040
- #define FEAT_FWUPDATE 0x080
- #define FEAT_HWLOAD 0x100
- #define FEAT_ANSI 0x200
- #define FEAT_ISO 0x400
- #define FEAT_MOUSEACCEL 0x800
- #define FEAT_COMMON (FEAT_BIND | FEAT_NOTIFY | FEAT_FWVERSION | FEAT_MOUSEACCEL | FEAT_HWLOAD)
- #define FEAT_STD_RGB (FEAT_COMMON | FEAT_RGB | FEAT_POLLRATE | FEAT_FWUPDATE)
- #define FEAT_STD_NRGB (FEAT_COMMON)
- #define FEAT_LMASK (FEAT_ANSI | FEAT_ISO)
- #define HAS_FEATURES(kb, feat) (((kb)->features & (feat)) == (feat))
- #define HAS_ANY_FEATURE(kb, feat) (!!((kb)->features & (feat)))
- #define NEEDS_FW_UPDATE(kb) ((kb)->fwversion == 0 && HAS_FEATURES((kb), FEAT_FWUPDATE | FEAT_FWVERSION))
- #define SCROLL_ACCELERATED 0
- #define SCROLL_MIN 1
- #define SCROLL_MAX 10
- #define KB_NAME_LEN 40
- #define SERIAL_LEN 34
- #define MSG_SIZE 64
- #define IFACE_MAX 4

Variables

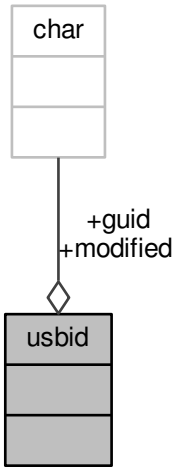
- const union devcmd vtable_keyboard
- const union devcmd vtable_keyboard_nonrgb
- const union devcmd vtable_mouse

8.40.1 Data Structure Documentation

8.40.1.1 struct usbid

Definition at line 8 of file structures.h.

Collaboration diagram for usbid:



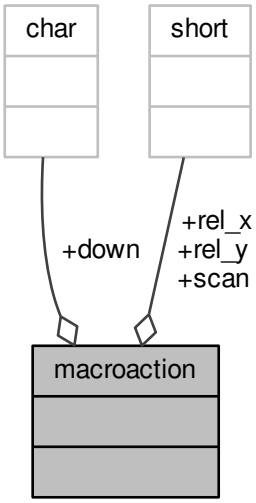
Data Fields

char	guid[16]	
char	modified[4]	

8.40.1.2 struct macroaction

Definition at line 27 of file structures.h.

Collaboration diagram for macroaction:



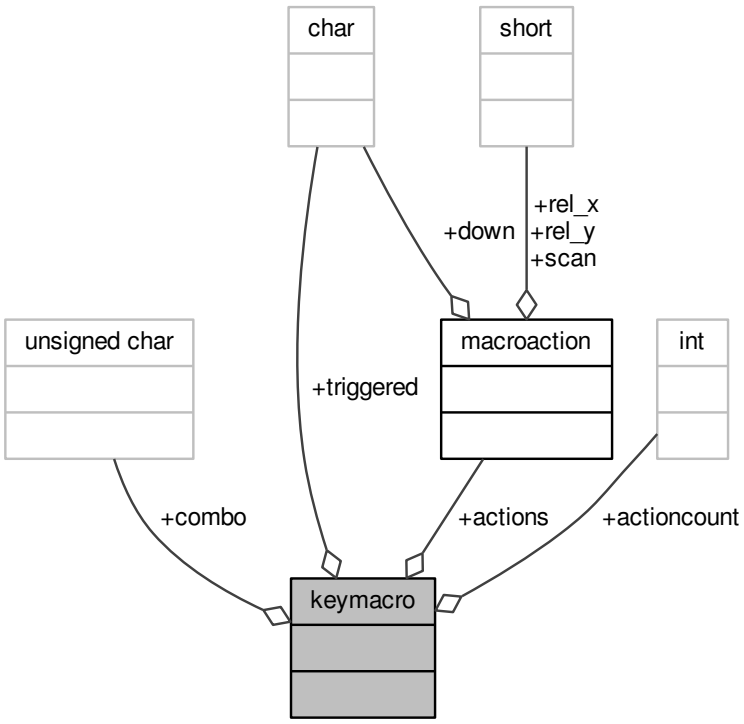
Data Fields

char	down	
short	rel_x	
short	rel_y	
short	scan	

8.40.1.3 struct keymacro

Definition at line 34 of file structures.h.

Collaboration diagram for keymacro:



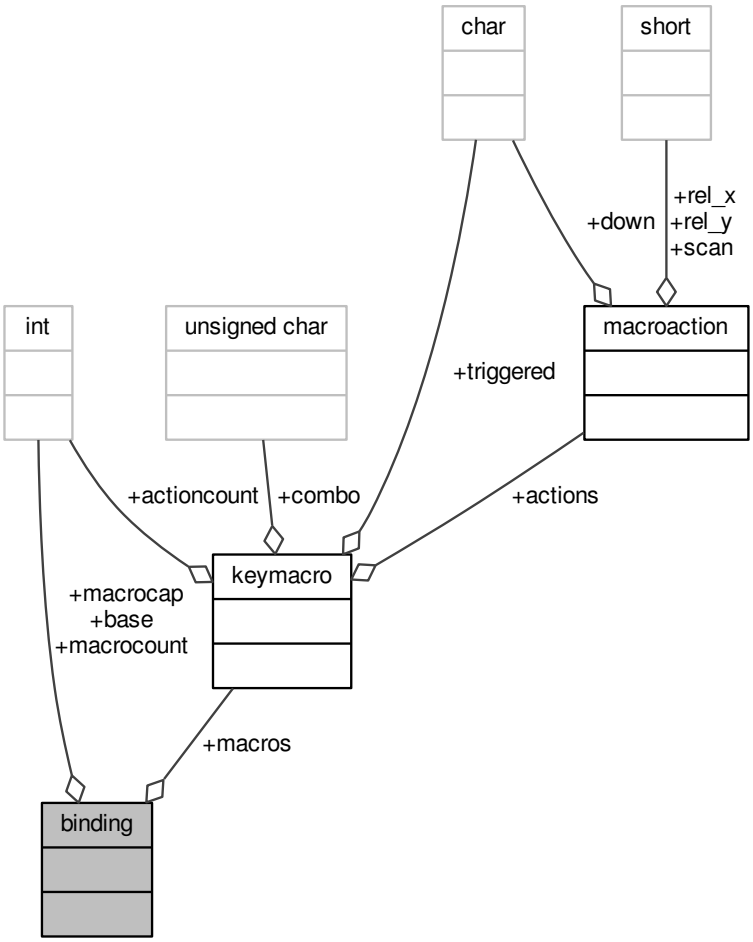
Data Fields

int	actioncount	
macroaction *	actions	
uchar	combo[(((((152+3+12)+25)+7)/8)]	
char	triggered	

8.40.1.4 struct binding

Definition at line 42 of file structures.h.

Collaboration diagram for binding:



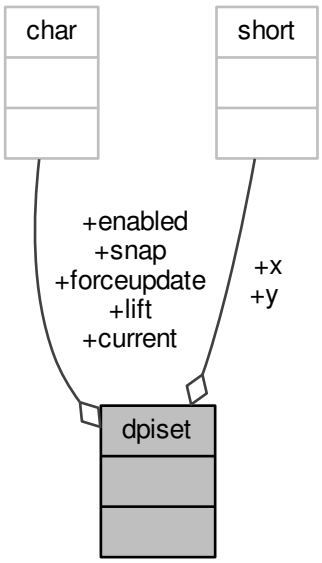
Data Fields

int	base[((152+3+12)+25)]	
int	macrocap	
int	macrocount	
keymacro *	macros	

8.40.1.5 struct dpiset

Definition at line 56 of file structures.h.

Collaboration diagram for dpiset:



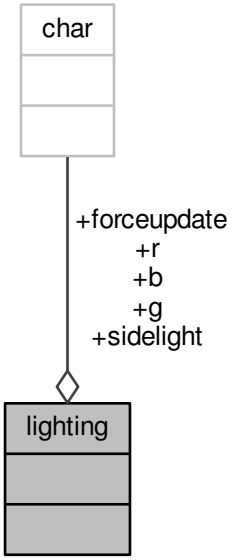
Data Fields

uchar	current	
uchar	enabled	
uchar	forceupdate	
uchar	lift	
uchar	snap	
ushort	x[6]	
ushort	y[6]	

8.40.1.6 struct lighting

Definition at line 72 of file structures.h.

Collaboration diagram for lighting:



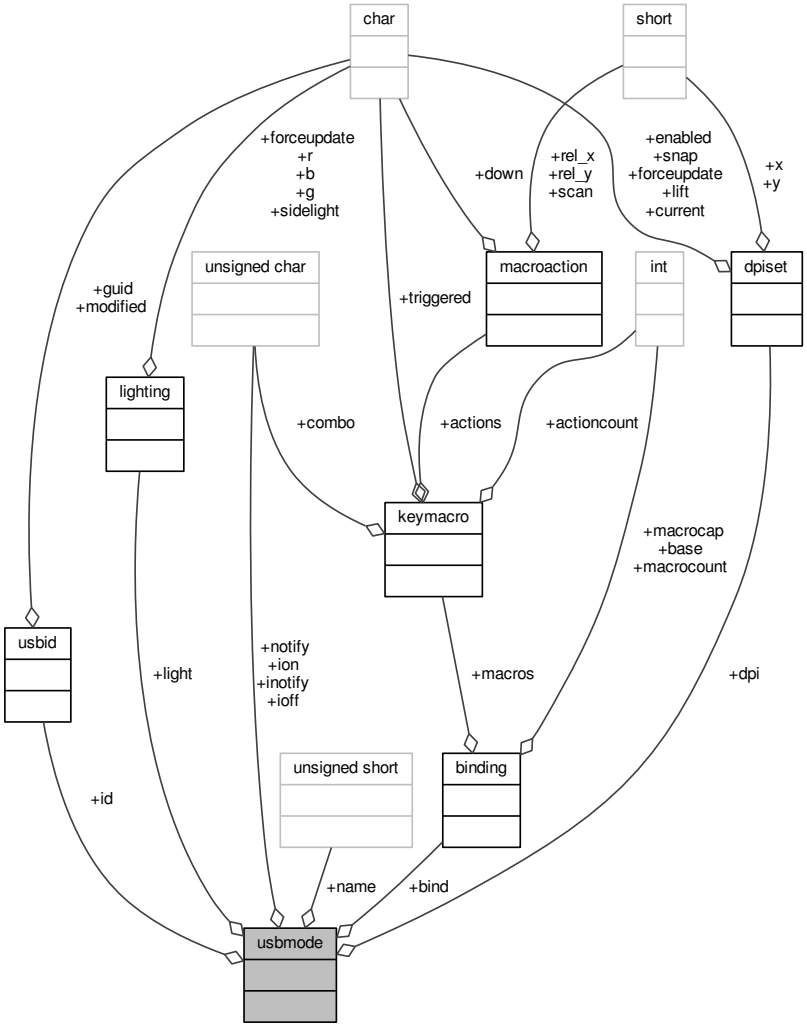
Data Fields

uchar	b[152+11]	
uchar	forceupdate	
uchar	g[152+11]	
uchar	r[152+11]	
uchar	sidelight	

8.40.1.7 struct usbmode

Definition at line 82 of file structures.h.

Collaboration diagram for usbmode:



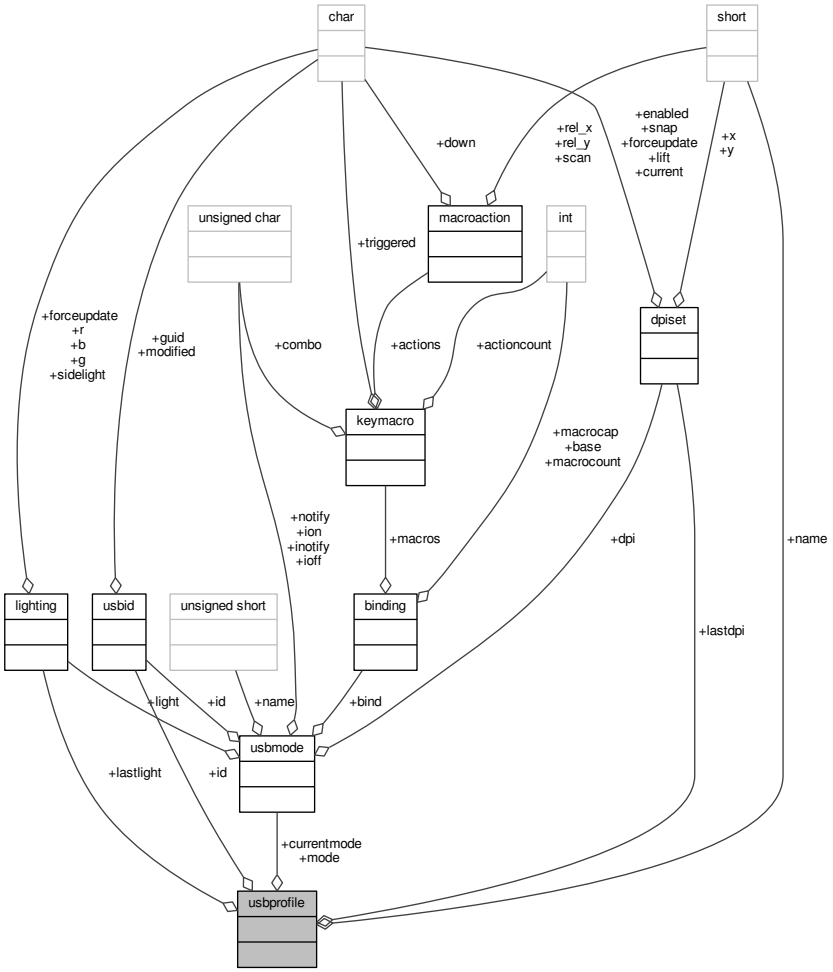
Data Fields

binding	bind	
dpiset	dpi	
usbid	id	
uchar	inotify[10]	
uchar	ioff	
uchar	ion	
lighting	light	
ushort	name[16]	
uchar	notify[10][((((152+3+12)+25)+7)/8)]	

8.40.1.8 struct usbprofile

Definition at line 100 of file structures.h.

Collaboration diagram for usbprofile:



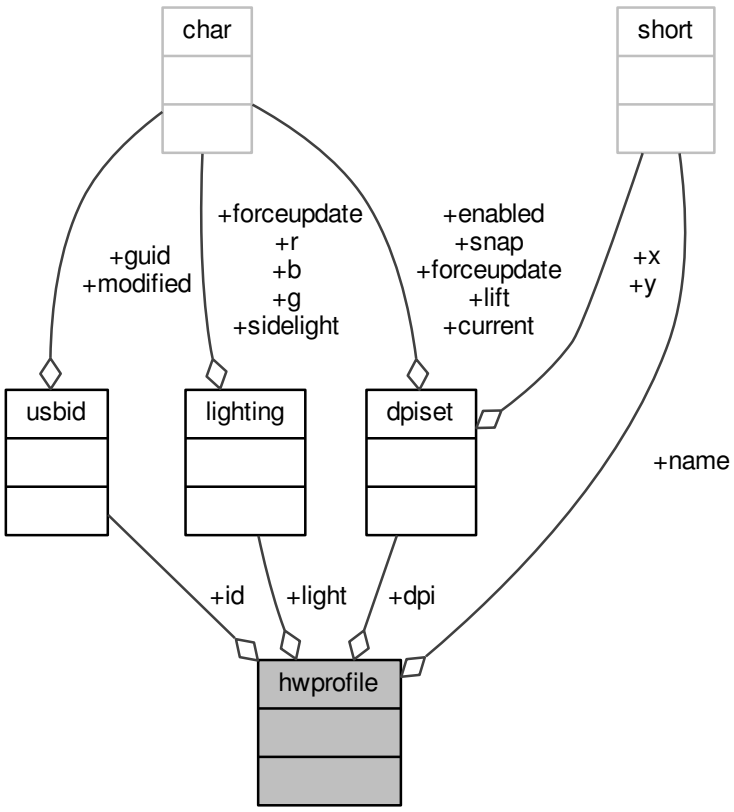
Data Fields

usbmode *	currentmode	
usbid	id	
dpiset	lastdpi	
lighting	lastlight	
usbmode	mode[6]	
ushort	name[16]	

8.40.1.9 struct hwprofile

Definition at line 117 of file structures.h.

Collaboration diagram for hwprofile:



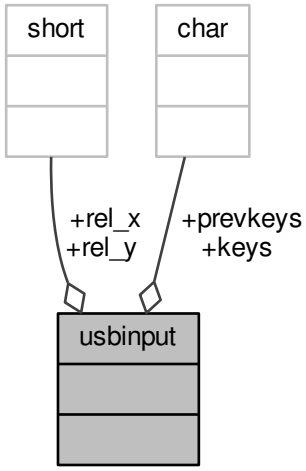
Data Fields

dpiset	dpi[3]	
usbid	id[3+1]	
lighting	light[3]	
ushort	name[3+1][16]	

8.40.1.10 struct usbinput

Definition at line 128 of file structures.h.

Collaboration diagram for usbinput:



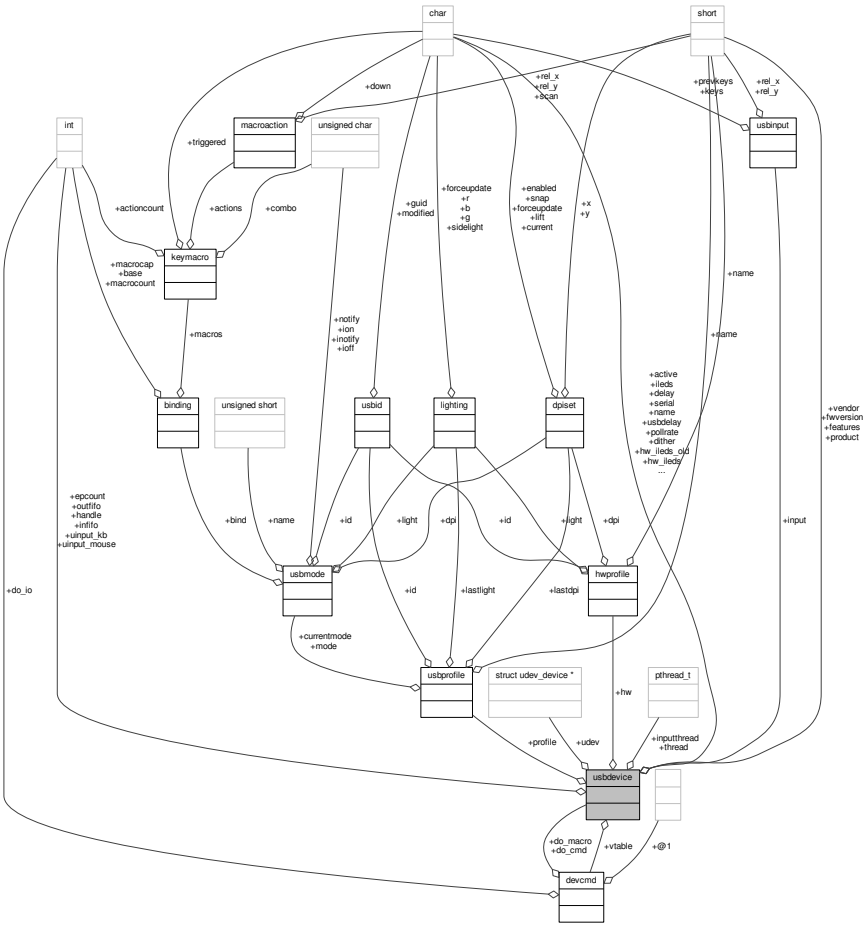
Data Fields

uchar	keys[(((((152+3+12)+25)+7)/8)]
uchar	prevkeys[(((((152+3+12)+25)+7)/8)]
short	rel_x
short	rel_y

8.40.1.11 struct usbdevice

Definition at line 177 of file structures.h.

Collaboration diagram for usbdevice:



Data Fields

char	active	
char	delay	
char	dither	
int	epcount	
ushort	features	
ushort	fwversion	
int	handle	
hwprofile *	hw	
uchar	hw_ileds	
uchar	hw_ileds_old	
uchar	ileds	
int	infifo	
usbinput	input	
pthread_t	inputthread	
char	name[40+1]	

int	outfifo[10]	
char	pollrate	
short	product	
usbprofile *	profile	
char	serial[34]	
pthread_t	thread	
struct udev_device *	udev	
int	uinput_kb	
int	uinput_mouse	
char	usbdelay	
short	vendor	
const union devcmd *	vtable	

8.40.2 Macro Definition Documentation

8.40.2.1 #define CLEAR_KEYBIT(*array, index*) do { (array)[(index) / 8] &= ~(1 << ((index) % 8)); } while(0)

Definition at line 16 of file structures.h.

Referenced by cmd_notify(), corsair_mousecopy(), hid_kb_translate(), and hid_mouse_translate().

8.40.2.2 #define DPI_COUNT 6

Definition at line 53 of file structures.h.

Referenced by cmd_dpi(), cmd_dpisel(), loaddpi(), printdpi(), savedpi(), and updatedpi().

8.40.2.3 #define FEAT_ADJRATE 0x008

Definition at line 138 of file structures.h.

Referenced by _mkdevpath(), _setupusb(), and _start_dev().

8.40.2.4 #define FEAT_ANSI 0x200

Definition at line 145 of file structures.h.

Referenced by readcmd().

8.40.2.5 #define FEAT_BIND 0x010

Definition at line 139 of file structures.h.

Referenced by _mkdevpath(), main(), and readcmd().

8.40.2.6 #define FEAT_COMMON (FEAT_BIND | FEAT_NOTIFY | FEAT_FWVERSION | FEAT_MOUSEACCEL | FEAT_HWLOAD)

Definition at line 150 of file structures.h.

8.40.2.7 #define FEAT_FWUPDATE 0x080

Definition at line 142 of file structures.h.

Referenced by _mkdevpath(), _start_dev(), and cmd_fwupdate().

8.40.2.8 #define FEAT_FWVERSION 0x040

Definition at line 141 of file structures.h.

Referenced by _mkdevpath(), and _start_dev().

8.40.2.9 #define FEAT_HWLOAD 0x100

Definition at line 143 of file structures.h.

Referenced by _start_dev().

8.40.2.10 #define FEAT_ISO 0x400

Definition at line 146 of file structures.h.

Referenced by readcmd().

8.40.2.11 #define FEAT_LMASK (FEAT_ANSI | FEAT_ISO)

Definition at line 153 of file structures.h.

Referenced by readcmd().

8.40.2.12 #define FEAT_MONOCHROME 0x002

Definition at line 136 of file structures.h.

Referenced by _mkdevpath(), and _setupusb().

8.40.2.13 #define FEAT_MOUSEACCEL 0x800

Definition at line 147 of file structures.h.

Referenced by main(), and readcmd().

8.40.2.14 #define FEAT_NOTIFY 0x020

Definition at line 140 of file structures.h.

Referenced by _mkdevpath(), main(), and readcmd().

8.40.2.15 #define FEAT_POLLRATE 0x004

Definition at line 137 of file structures.h.

Referenced by _mkdevpath(), _start_dev(), and getfwversion().

8.40.2.16 #define FEAT_RGB 0x001

Definition at line 135 of file structures.h.

Referenced by _mkdevpath(), _start_dev(), os_setupusb(), revertusb(), and usbunclaim().

8.40.2.17 #define FEAT_STD_NRGB (FEAT_COMMON)

Definition at line 152 of file structures.h.

Referenced by _setupusb().

8.40.2.18 #define FEAT_STD_RGB (FEAT_COMMON | FEAT_RGB | FEAT_POLLRATE | FEAT_FWUPDATE)

Definition at line 151 of file structures.h.

Referenced by _setupusb().

8.40.2.19 #define HAS_ANY_FEATURE(*kb, feat*) (!!((kb)->features & (feat)))

Definition at line 157 of file structures.h.

8.40.2.20 #define HAS_FEATURES(*kb, feat*) (((kb)->features & (feat)) == (feat))

Definition at line 156 of file structures.h.

Referenced by _mkdevpath(), _start_dev(), cmd_fwupdate(), os_setupusb(), readcmd(), revertusb(), and usbunclaim().

8.40.2.21 #define HWMODE_K70 1

Definition at line 114 of file structures.h.

Referenced by cmd_hwload_kb(), and cmd_hwsave_kb().

8.40.2.22 #define HWMODE_K95 3

Definition at line 115 of file structures.h.

Referenced by cmd_hwload_kb(), and cmd_hwsave_kb().

8.40.2.23 #define HWMODE_MAX 3

Definition at line 116 of file structures.h.

8.40.2.24 #define I_CAPS 2

Definition at line 20 of file structures.h.

Referenced by _cmd_get(), iselect(), nprintind(), and updateindicators_kb().

8.40.2.25 #define I_NUM 1

Definition at line 19 of file structures.h.

Referenced by _cmd_get(), iselect(), nprintind(), and updateindicators_kb().

8.40.2.26 #define I_SCROLL 4

Definition at line 21 of file structures.h.

Referenced by _cmd_get(), iselect(), nprintind(), and updateindicators_kb().

8.40.2.27 #define IFACE_MAX 4

Definition at line 176 of file structures.h.

8.40.2.28 #define KB_NAME_LEN 40

Definition at line 173 of file structures.h.

Referenced by _setupusb(), and os_setupusb().

8.40.2.29 #define LIFT_MAX 5

Definition at line 55 of file structures.h.

Referenced by cmd_lift(), and loaddpi().

8.40.2.30 #define LIFT_MIN 1

Definition at line 54 of file structures.h.

Referenced by cmd_lift(), and loaddpi().

8.40.2.31 #define MACRO_MAX 1024

Definition at line 50 of file structures.h.

Referenced by _cmd_macro().

8.40.2.32 #define MD_NAME_LEN 16

Definition at line 81 of file structures.h.

Referenced by cmd_hwsave_kb(), cmd_hwsave_mouse(), cmd_name(), gethwmodename(), gethwprofilename(), getmodename(), hwloadmode(), hwtonative(), and nativetohw().

8.40.2.33 #define MODE_COUNT 6

Definition at line 99 of file structures.h.

Referenced by _freeprofile(), allocprofile(), and readcmd().

8.40.2.34 #define MSG_SIZE 64

Definition at line 175 of file structures.h.

Referenced by _usbsend(), cmd_hwload_kb(), cmd_hwload_mouse(), cmd_hwsave_kb(), cmd_hwsave_mouse(), cmd_pollrate(), fwupdate(), getfwversion(), hwloadmode(), loaddpi(), loadrgb_kb(), loadrgb_mouse(), os_inputmain(), os_usbrecv(), os_usbsend(), savedpi(), savergb_kb(), savergb_mouse(), setactive_kb(), setactive_mouse(), updatedpi(), updatergb_kb(), and updatergb_mouse().

8.40.2.35 #define NEEDS_FW_UPDATE(*kb*) ((kb)->fwversion == 0 && HAS_FEATURES((kb), FEAT_FWUPDATE | FEAT_FWVERSION))

Definition at line 160 of file structures.h.

Referenced by _start_dev(), readcmd(), revertusb(), setactive_kb(), and setactive_mouse().

8.40.2.36 #define OUTFIFO_MAX 10

Definition at line 24 of file structures.h.

Referenced by _mknotifynode(), _rmnotifynode(), inputupdate_keys(), nprintf(), readcmd(), rmdevpath(), and updateindicators_kb().

8.40.2.37 #define PR_NAME_LEN 16

Definition at line 98 of file structures.h.

Referenced by cmd_hwload_kb(), cmd_hwload_mouse(), cmd_profilename(), getprofilename(), hwtonative(), and nativetohw().

8.40.2.38 #define SCROLL_ACCELERATED 0

Definition at line 163 of file structures.h.

Referenced by readcmd().

8.40.2.39 #define SCROLL_MAX 10

Definition at line 165 of file structures.h.

Referenced by readcmd().

8.40.2.40 #define SCROLL_MIN 1

Definition at line 164 of file structures.h.

Referenced by readcmd().

8.40.2.41 #define SERIAL_LEN 34

Definition at line 174 of file structures.h.

Referenced by _setupusb(), and os_setupusb().

8.40.2.42 #define SET_KEYBIT(*array, index*) do { (array)[(index) / 8] |= 1 << ((index) % 8); } while(0)

Definition at line 15 of file structures.h.

Referenced by _cmd_macro(), cmd_notify(), corsair_mousecopy(), hid_kb_translate(), and hid_mouse_translate().

8.40.3 Variable Documentation

8.40.3.1 const union devcmd vtable_keyboard

Definition at line 28 of file device_vtable.c.

Referenced by get_vtable().

8.40.3.2 const union devcmd vtable_keyboard_nonrgb

Definition at line 75 of file device_vtable.c.

Referenced by get_vtable().

8.40.3.3 const union devcmd vtable_mouse

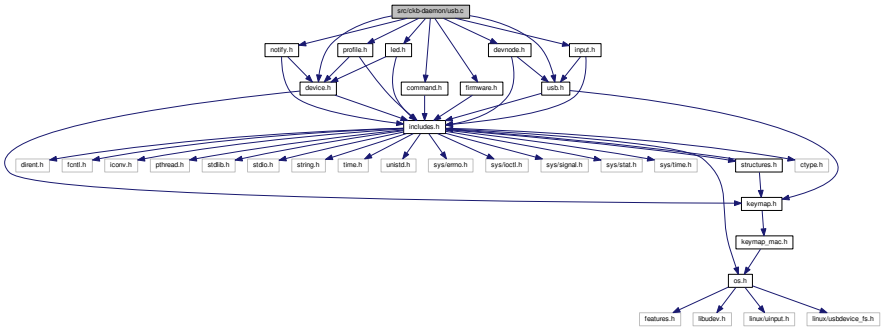
Definition at line 122 of file device_vtable.c.

Referenced by get_vtable().

8.41 src/ckb-daemon/usb.c File Reference

```
#include "command.h"
#include "device.h"
#include "devnode.h"
#include "firmware.h"
#include "input.h"
#include "led.h"
#include "notify.h"
#include "profile.h"
#include "usb.h"
```

Include dependency graph for usb.c:



Functions

- const char * vendor_str (short vendor)
- const char * product_str (short product)
- static const devcmd * get_vtable (short vendor, short product)
- static void * devmain (usbdevice *kb)
- static void * _setupusb (void *context)
- void setupusb (usbdevice *kb)
- int revertusb (usbdevice *kb)
- int _resetusb (usbdevice *kb, const char *file, int line)
- int usb_tryreset (usbdevice *kb)
- int _usbsend (usbdevice *kb, const uchar *messages, int count, const char *file, int line)
- int _usbrecv (usbdevice *kb, const uchar *out_msg, uchar *in_msg, const char *file, int line)
- int closeusb (usbdevice *kb)

Variables

- pthread_mutex_t usbmutex = PTHREAD_MUTEX_INITIALIZER
- volatile int reset_stop = 0
- int features_mask = -1
- int hwload_mode

8.41.1 Function Documentation

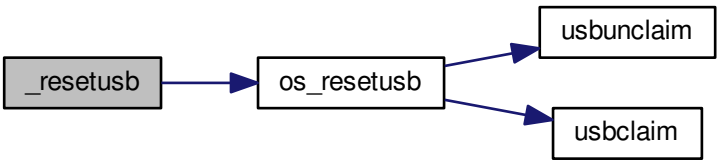
8.41.1.1 int _resetusb (usbdevice * *kb*, const char * *file*, int *line*)

Definition at line 149 of file usb.c.

References usbdevice::active, DELAY_LONG, os_resetusb(), and usbdevice::vtable.

```
149                                                         {
150     // Perform a USB reset
151     DELAY_LONG(kb);
152     int res = os_resetusb(kb, file, line);
153     if(res)
154         return res;
155     DELAY_LONG(kb);
156     // Re-initialize the device
157     if(kb->vtable->start(kb, kb->active) != 0)
158         return -1;
159     if(kb->vtable->updatergb(kb, 1) != 0)
160         return -1;
161     return 0;
162 }
```

Here is the call graph for this function:



8.41.1.2 static void* _setupusb (void * *context*) [static]

Definition at line 77 of file usb.c.

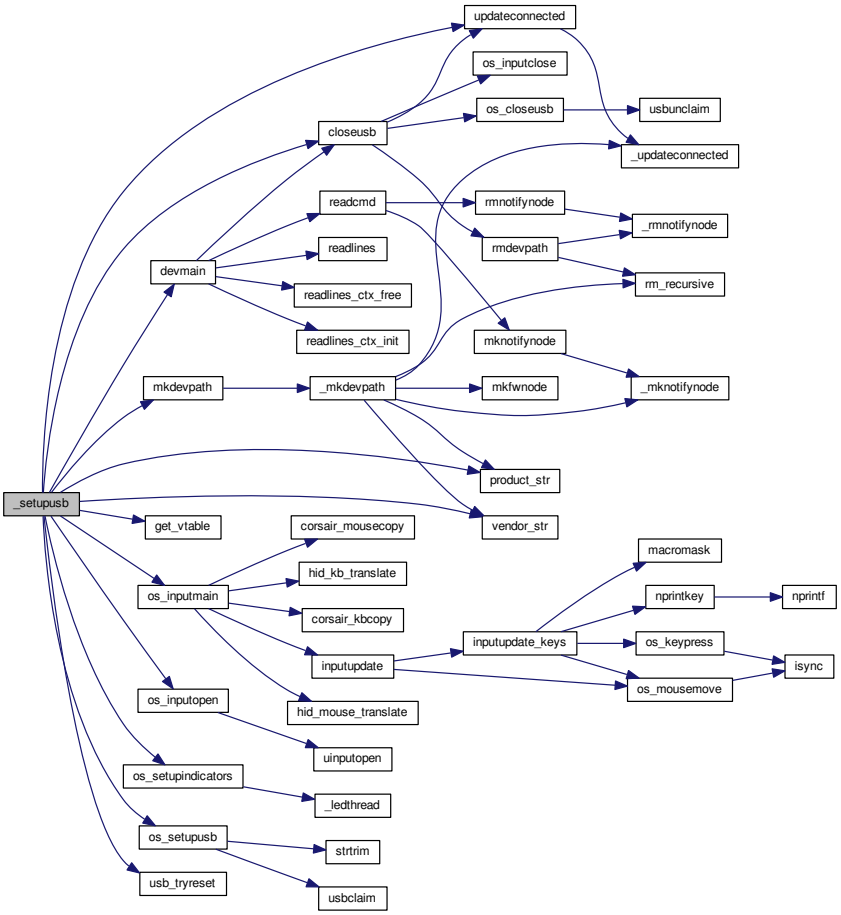
References ckb_info, closeusb(), DELAY_LONG, devmain(), devpath, dmutex, FEAT_ADJRATE, FEAT_MONOCHROME, FEAT_STD_NRGB, FEAT_STD_RGB, usbdevice::features, features_mask, get_vtable(), imutex, INDEX_OF, usbdevice::inputthread, IS_MONOCHROME, IS_MOUSE, IS_RGB, KB_NAME_LEN, keyboard, mkdevpath(), usbdevice::name, os_inputmain(), os_inputopen(), os_setupindicators(), os_setupusb(), usbdevice::product, product_str(), usbdevice::serial, SERIAL_LEN, updateconnected(), USB_DELAY_DEFAULT, usb_tryreset(), usbdevice::usbdelay, usbdevice::vendor, vendor_str(), and usbdevice::vtable.

Referenced by setupusb().

```
77                                      {
78     usbdevice* kb = context;
79     // Set standard fields
80     short vendor = kb->vendor, product = kb->product;
81     const devcmd* vt = kb->vtable = get_vtable(vendor, product);
82     kb->features = (IS_RGB(vendor, product) ? FEAT_STD_RGB :
       FEAT_STD_NRGB) & features_mask;
83     if(IS_MOUSE(vendor, product)) kb->features |= FEAT_ADJRATE;
84     if(IS_MONOCHROME(vendor, product)) kb->features |=
       FEAT_MONOCHROME;
85     kb->usbdelay = USB_DELAY_DEFAULT;
86
87     // Perform OS-specific setup
88     DELAY_LONG(kb);
89     if(os_setupusb(kb))
90         goto fail;
```

```
91      // Make up a device name and serial if they weren't assigned
92      if(!kb->serial[0])
93          snprintf(kb->serial, SERIAL_LEN, "%04x:%04x-NoID", kb->
        vendor, kb->product);
94      if(!kb->name[0])
95          snprintf(kb->name, KB_NAME_LEN, "%s %s", vendor_str(kb->
        vendor), product_str(kb->product));
96
97      // Set up an input device for key events
98      if(os_inputopen(kb))
99          goto fail;
100     if(pthread_create(&kb->inputthread, 0, os_inputmain, kb))
101         goto fail;
102     pthread_detach(kb->inputthread);
103     if(os_setupindicators(kb))
104         goto fail;
105
106     // Set up device
107     vt->allocprofile(kb);
108     vt->updateindicators(kb, 1);
109     pthread_mutex_unlock(imutex(kb));
110     if(vt->start(kb, 0) && usb_tryreset(kb))
111         goto fail_noinput;
112
113     // Make /dev path
114     if(mkdevpath(kb))
115         goto fail_noinput;
116
117     // Finished. Enter main loop
118     int index = INDEX_OF(kb, keyboard);
119     ckb_info("Setup finished for %s%d\n", devpath, index);
120     updateconnected();
121     return devmain(kb);
122
123     fail:
124     pthread_mutex_unlock(imutex(kb));
125     fail_noinput:
126     closeusb(kb);
127     pthread_mutex_unlock(dmutex(kb));
128     return 0;
129 }
```

Here is the call graph for this function:



Here is the caller graph for this function:

8.41.1.3 int _usbrecv (usbdevice * *kb*, const uchar * *out_msg*, uchar * *in_msg*, const char * *file*, int *line*)

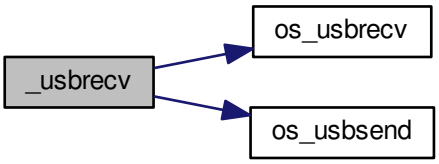
Definition at line 207 of file usb.c.

References ckb_err_fn, DELAY_LONG, DELAY_MEDIUM, DELAY_SHORT, hwload_mode, os_usbrecv(), os_usbsend(), and reset_stop.

```
207                                                                                                 {
208     // Try a maximum of 3 times
209     for(int try = 0; try < 5; try++){
210         // Send the output message
211         DELAY_SHORT(kb);
212         int res = os_usbsend(kb, out_msg, 1, file, line);
213         if(res == 0)
214             return 0;
215         else if(res == -1){
```

```
216             // Retry on temporary failure
217             if(reset_stop)
218                 return 0;
219             DELAY_LONG(kb);
220             continue;
221         }
222         // Wait for the response
223         DELAY_MEDIUM(kb);
224         res = os_usbrecv(kb, in_msg, file, line);
225         if(res == 0)
226             return 0;
227         else if(res != -1)
228             return res;
229         if(reset_stop || hwload_mode != 2)
230             return 0;
231         DELAY_LONG(kb);
232     }
233     // Give up
234     ckb_err_fn("Too many send/recv failures. Dropping.\n", file, line);
235     return 0;
236 }
```

Here is the call graph for this function:



8.41.1.4 int _usbsend (usbdevice * *kb*, const uchar * *messages*, int *count*, const char * *file*, int *line*)

Definition at line 184 of file usb.c.

References DELAY_LONG, DELAY_SHORT, hwload_mode, MSG_SIZE, os_usbsend(), and reset_stop.

```
184                                                                                                 {
185     int total_sent = 0;
186     for(int i = 0; i < count; i++){
187         // Send each message via the OS function
188         while(1){
189             DELAY_SHORT(kb);
190             int res = os_usbsend(kb, messages + i * MSG_SIZE, 0, file, line);
191             if(res == 0)
192                 return 0;
193             else if(res != -1){
194                 total_sent += res;
195                 break;
196             }
197             // Stop immediately if the program is shutting down or hardware load is set to tryonce
198             if(reset_stop || hwload_mode != 2)
199                 return 0;
200             // Retry as long as the result is temporary failure
201             DELAY_LONG(kb);
202         }
203     }
204     return total_sent;
205 }
```

Here is the call graph for this function:

8.41.1.5 int closeusb (usbdevice * *kb*)

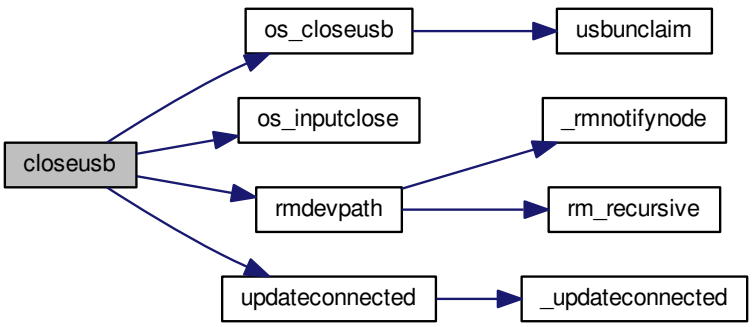
Definition at line 238 of file usb.c.

References ckb_info, devpath, dmutex, usbdevice::handle, imutex, INDEX_OF, keyboard, os_closeusb(), os_inputclose(), rmdevpath(), usbdevice::thread, updateconnected(), and usbdevice::vtable.

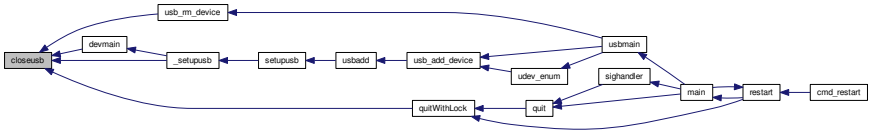
Referenced by _setupusb(), devmain(), quitWithLock(), and usb_rm_device().

```
238                             {
239     pthread_mutex_lock(imutex(kb));
240     if(kb->handle){
241         int index = INDEX_OF(kb, keyboard);
242         ckb_info("Disconnecting %s%d\n", devpath, index);
243         os_inputclose(kb);
244         updateconnected();
245         // Close USB device
246         os_closeusb(kb);
247     } else
248         updateconnected();
249     rmdevpath(kb);
250
251     // Wait for thread to close
252     pthread_mutex_unlock(imutex(kb));
253     pthread_mutex_unlock(dmutex(kb));
254     pthread_join(kb->thread, 0);
255     pthread_mutex_lock(dmutex(kb));
256
257     // Delete the profile and the control path
258     if(!kb->vtable)
259         return 0;
260     kb->vtable->freeprofile(kb);
261     memset(kb, 0, sizeof(usbdevice));
262     return 0;
263 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



8.41.1.6 static void* devmain (usbdevice * *kb*) [static]

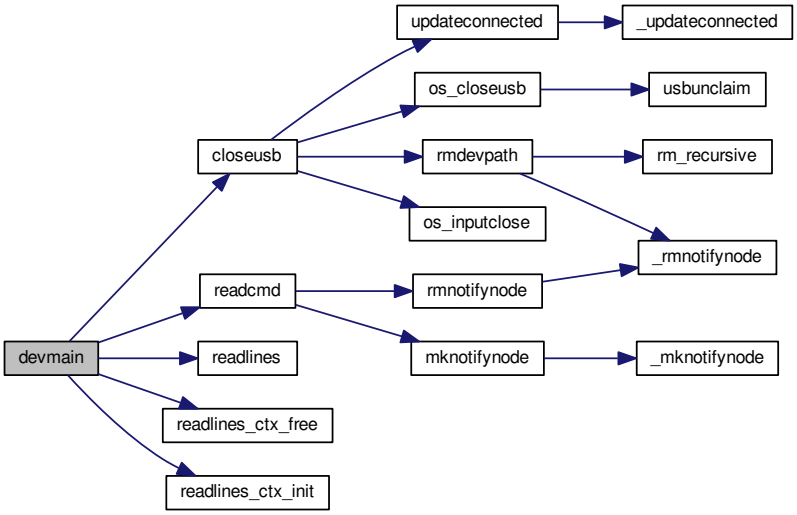
Definition at line 50 of file usb.c.

References closeusb(), dmutex, usbdevice::infifo, IS_CONNECTED, readcmd(), readlines(), readlines_ctx_free(), and readlines_ctx_init().

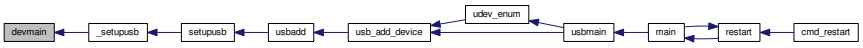
Referenced by _setupusb().

```
50                                  {
51     // dmutex should still be locked when this is called
52     int kbfifo = kb->infifo - 1;
53     readlines_ctx linectx;
54     readlines_ctx_init(&linectx);
55     while(1){
56         pthread_mutex_unlock(dmutex(kb));
57         // Read from FIFO
58         const char* line;
59         int lines = readlines(kbfifo, linectx, &line);
60         pthread_mutex_lock(dmutex(kb));
61         // End thread when the handle is removed
62         if(!IS_CONNECTED(kb))
63             break;
64         if(lines){
65             if(readcmd(kb, line)){
66                 // USB transfer failed; destroy device
67                 closeusb(kb);
68                 break;
69             }
70         }
71     }
72     pthread_mutex_unlock(dmutex(kb));
73     readlines_ctx_free(linectx);
74     return 0;
75 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



8.41.1.7 static const devcmd* get_vtable (short *vendor,* short *product*) [static]

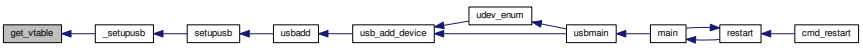
Definition at line 45 of file usb.c.

References IS_MOUSE, IS_RGB, vtable_keyboard, vtable_keyboard_nonrgb, and vtable_mouse.

Referenced by _setupusb().

```
45                                                          {
46      return IS_MOUSE(vendor, product) ? &vtable_mouse :
       IS_RGB(vendor, product) ? &vtable_keyboard : &
       vtable_keyboard_nonrgb;
47 }
```

Here is the caller graph for this function:



8.41.1.8 const char* product_str (short *product*)

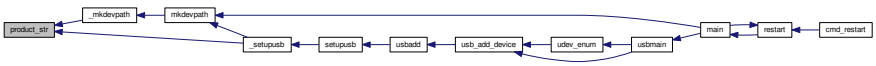
Definition at line 26 of file usb.c.

References P_K65, P_K65_LUX, P_K65_NRGB, P_K65_RFIRE, P_K70, P_K70_LUX, P_K70_LUX_NRGB, P_K70_NRGB, P_K70_RFIRE, P_K70_RFIRE_NRGB, P_K95, P_K95_NRGB, P_K95_PLATINUM, P_M65, P_M65_PRO, P_SABRE_L, P_SABRE_N, P_SABRE_O, P_SABRE_O2, P_SCIMITAR, P_SCIMITAR_PRO, P_STRAFE, and P_STRAFE_NRGB.

Referenced by _mkdevpath(), and _setupusb().

```
26                                      {
27     if(product == P_K95 || product == P_K95_NRGB || product ==
       P_K95_PLATINUM)
28         return "k95";
29     if(product == P_K70 || product == P_K70_NRGB || product ==
       P_K70_LUX || product == P_K70_LUX_NRGB || product ==
       P_K70_RFIRE || product == P_K70_RFIRE_NRGB)
30         return "k70";
31     if(product == P_K65 || product == P_K65_NRGB || product ==
       P_K65_LUX || product == P_K65_RFIRE)
32         return "k65";
33     if(product == P_STRAFE || product == P_STRAFE_NRGB)
34         return "strafe";
35     if(product == P_M65 || product == P_M65_PRO)
36         return "m65";
37     if(product == P_SABRE_O || product == P_SABRE_L || product ==
       P_SABRE_N || product == P_SABRE_O2)
38         return "sabre";
39     if(product == P_SCIMITAR || product == P_SCIMITAR_PRO)
40         return "scimitar";
41     return "";
42 }
```

Here is the caller graph for this function:



8.41.1.9 int revertusb (usbdevice * *kb*)

Definition at line 137 of file usb.c.

References FEAT_RGB, HAS_FEATURES, NEEDS_FW_UPDATE, NK95_HWON, nk95cmd, and setactive.

Referenced by quitWithLock().

```
137                                {
138     if(NEEDS_FW_UPDATE(kb))
139         return 0;
140     if(!HAS_FEATURES(kb, FEAT_RGB)){
141         nk95cmd(kb, NK95_HWON);
142         return 0;
143     }
144     if(setactive(kb, 0))
145         return -1;
146     return 0;
147 }
```

Here is the caller graph for this function:

8.41.1.10 void setupusb (usbdevice * kb)

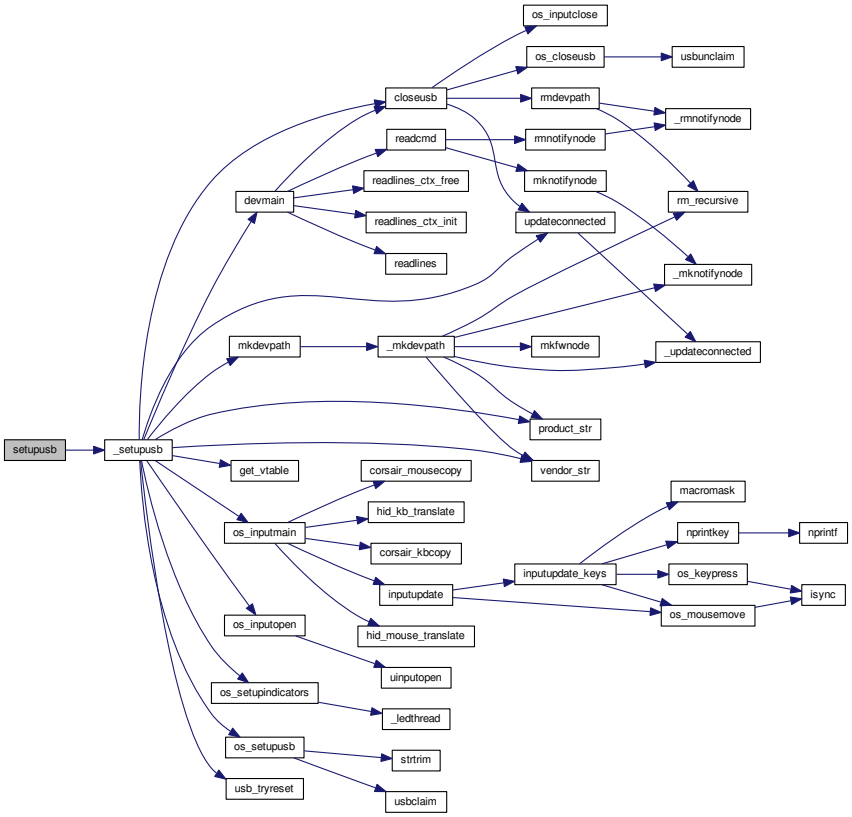
Definition at line 131 of file usb.c.

References _setupusb(), ckb_err, imutex, and usbdevice::thread.

Referenced by usbadd().

```
131                             {
132     pthread_mutex_lock(imutex(kb));
133     if(pthread_create(&kb->thread, 0, _setupusb, kb))
134         ckb_err("Failed to create USB thread\n");
135 }
```

Here is the call graph for this function:



Here is the caller graph for this function:

8.41.1.11 int usb_tryreset (usbdevice * kb)

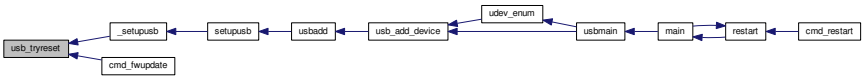
Definition at line 164 of file usb.c.

References ckb_err, ckb_info, reset_stop, and resetusb.

Referenced by _setupusb(), and cmd_fwupdate().

```
164                                 {
165     if(reset_stop)
166         return -1;
167     ckb_info("Attempting reset...\n");
168     while(1){
169         int res = resetusb(kb);
170         if(!res){
171             ckb_info("Reset success\n");
172             return 0;
173         }
174         if(res == -2 || reset_stop)
175             break;
176     }
177     ckb_err("Reset failed. Disconnecting.\n");
178     return -1;
179 }
```

Here is the caller graph for this function:



8.41.1.12 const char* vendor_str (short *vendor*)

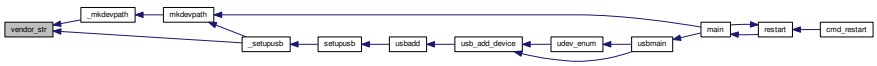
Definition at line 20 of file usb.c.

References V_CORSAIR.

Referenced by _mkdevpath(), and _setupusb().

```
20                                      {
21     if(vendor == V_CORSAIR)
22         return "corsair";
23     return "";
24 }
```

Here is the caller graph for this function:



8.41.2 Variable Documentation

8.41.2.1 int features_mask = -1

Definition at line 17 of file usb.c.

Referenced by _setupusb(), and main().

8.41.2.2 int hwload_mode

Definition at line 7 of file device.c.

Referenced by _start_dev(), _usbrecv(), and _usbsend().

8.41.2.3 volatile int reset_stop = 0

Definition at line 14 of file usb.c.

Referenced by _usbrecv(), _usbsend(), quitWithLock(), and usb_tryreset().

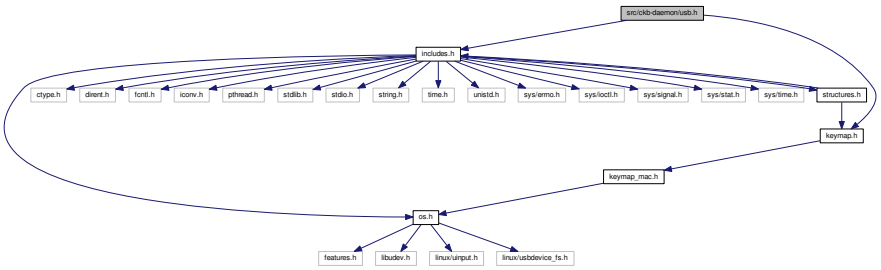
8.41.2.4 pthread_mutex_t usbmutex = PTHREAD_MUTEX_INITIALIZER

Definition at line 11 of file usb.c.

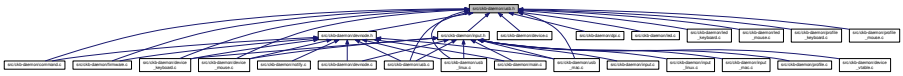
8.42 src/ckb-daemon/usb.h File Reference

```
#include "includes.h"
#include "keymap.h"
```

Include dependency graph for usb.h:



This graph shows which files directly or indirectly include this file:



Macros

- #define V_CORSAIR 0x1b1c
- #define V_CORSAIR_STR "1b1c"
- #define P_K65 0x1b17
- #define P_K65_STR "1b17"
- #define P_K65_NRGB 0x1b07
- #define P_K65_NRGB_STR "1b07"
- #define P_K65_LUX 0x1b37
- #define P_K65_LUX_STR "1b37"
- #define P_K65_RFIRE 0x1b39
- #define P_K65_RFIRE_STR "1b39"
- #define IS_K65(kb) ((kb)->vendor == V_CORSAIR && ((kb)->product == P_K65 || (kb)->product == P_K65_NRGB || (kb)->product == P_K65_LUX || (kb)->product == P_K65_RFIRE))
- #define P_K70 0x1b13
- #define P_K70_STR "1b13"
- #define P_K70_NRGB 0x1b09
- #define P_K70_NRGB_STR "1b09"
- #define P_K70_LUX 0x1b33

- #define P_K70_LUX_STR "1b33"
- #define P_K70_LUX_NRGB 0x1b36
- #define P_K70_LUX_NRGB_STR "1b36"
- #define P_K70_RFIRE 0x1b38
- #define P_K70_RFIRE_STR "1b38"
- #define P_K70_RFIRE_NRGB 0x1b3a
- #define P_K70_RFIRE_NRGB_STR "1b3a"
- #define IS_K70(kb) ((kb)->vendor == V_CORSAIR && ((kb)->product == P_K70 || (kb)->product == P_K70_NRGB || (kb)->product == P_K70_RFIRE || (kb)->product == P_K70_RFIRE_NRGB || (kb)->product == P_K70_LUX || (kb)->product == P_K70_LUX_NRGB))
- #define P_K95 0x1b11
- #define P_K95_STR "1b11"
- #define P_K95_NRGB 0x1b08
- #define P_K95_NRGB_STR "1b08"
- #define P_K95_PLATINUM 0x1b2d
- #define P_K95_PLATINUM_STR "1b2d"
- #define IS_K95(kb) ((kb)->vendor == V_CORSAIR && ((kb)->product == P_K95 || (kb)->product == P_K95_NRGB || (kb)->product == P_K95_PLATINUM))
- #define P_STRAFE 0x1b20
- #define P_STRAFE_STR "1b20"
- #define P_STRAFE_NRGB 0x1b15
- #define P_STRAFE_NRGB_STR "1b15"
- #define IS_STRAFE(kb) ((kb)->vendor == V_CORSAIR && ((kb)->product == P_STRAFE || (kb)->product == P_STRAFE_NRGB))
- #define P_M65 0x1b12
- #define P_M65_STR "1b12"
- #define P_M65_PRO 0x1b2e
- #define P_M65_PRO_STR "1b2e"
- #define IS_M65(kb) ((kb)->vendor == V_CORSAIR && ((kb)->product == P_M65 || (kb)->product == P_M65_PRO))
- #define P_SABRE_O 0x1b14 /* optical */
- #define P_SABRE_O_STR "1b14"
- #define P_SABRE_L 0x1b19 /* laser */
- #define P_SABRE_L_STR "1b19"
- #define P_SABRE_N 0x1b2f /* new? */
- #define P_SABRE_N_STR "1b2f"
- #define P_SABRE_O2 0x1b32 /* Observed on a CH-9000111-EU model SABRE */
- #define P_SABRE_O2_STR "1b32"
- #define IS_SABRE(kb) ((kb)->vendor == V_CORSAIR && ((kb)->product == P_SABRE_O || (kb)->product == P_SABRE_L || (kb)->product == P_SABRE_N || (kb)->product == P_SABRE_O2))
- #define P_SCIMITAR 0x1b1e
- #define P_SCIMITAR_STR "1b1e"
- #define P_SCIMITAR_PRO 0x1b3e
- #define P_SCIMITAR_PRO_STR "1b3e"
- #define IS_SCIMITAR(kb) ((kb)->vendor == V_CORSAIR && ((kb)->product == P_SCIMITAR || (kb)->product == P_SCIMITAR_PRO))
- #define IS_RGB(vendor, product) ((vendor) == (V_CORSAIR) && (product) != (P_K65_NRGB) && (product) != (P_K70_NRGB) && (product) != (P_K95_NRGB))
- #define IS_MONOCHROME(vendor, product) ((vendor) == (V_CORSAIR) && (product) == (P_STRAFE_NRGB))
- #define IS_RGB_DEV(kb) IS_RGB((kb)->vendor, (kb)->product)
- #define IS_MONOCHROME_DEV(kb) IS_MONOCHROME((kb)->vendor, (kb)->product)
- #define IS_FULLRANGE(kb) (IS_RGB((kb)->vendor, (kb)->product) && (kb)->product != P_K65 && (kb)->product != P_K70 && (kb)->product != P_K95)

- #define IS_MOUSE(vendor, product) ((vendor) == (V_CORSAIR) && ((product) == (P_M65) || (product) == (P_M65_PRO) || (product) == (P_SABRE_O) || (product) == (P_SABRE_L) || (product) == (P_SABRE_N) || (product) == (P_SCIMITAR) || (product) == (P_SCIMITAR_PRO) || (product) == (P_SABRE_O2)))
- #define IS_MOUSE_DEV(kb) IS_MOUSE((kb)->vendor, (kb)->product)
- #define DELAY_SHORT(kb) usleep((int)(kb)->usbdelay * 1000)
- #define DELAY_MEDIUM(kb) usleep((int)(kb)->usbdelay * 10000)
- #define DELAY_LONG(kb) usleep(100000)
- #define USB_DELAY_DEFAULT 5
- #define resetusb(kb) _resetusb(kb, __FILE_NOPATH__, __LINE__)
- #define usbsend(kb, messages, count) _usbsend(kb, messages, count, __FILE_NOPATH__, __LINE__)
- #define usbrecv(kb, out_msg, in_msg) _usbrecv(kb, out_msg, in_msg, __FILE_NOPATH__, __LINE__)
- #define nk95cmd(kb, command) _nk95cmd(kb, (command) >> 16 & 0xFF, (command) & 0xFFFF, __FILE_NOPATH__, __LINE__)
- #define NK95_HWOFF 0x020030
- #define NK95_HWON 0x020001
- #define NK95_M1 0x140001
- #define NK95_M2 0x140002
- #define NK95_M3 0x140003

Functions

- const char * vendor_str (short vendor)
- const char * product_str (short product)
- int usbmain ()
- void usbkill ()
- void setupusb (usbdevice *kb)
- int os_setupusb (usbdevice *kb)
- void * os_inputmain (void *kb)
- int revertusb (usbdevice *kb)
- int closeusb (usbdevice *kb)
- void os_closeusb (usbdevice *kb)
- int _resetusb (usbdevice *kb, const char *file, int line)
- int os_resetusb (usbdevice *kb, const char *file, int line)
- int _usbsend (usbdevice *kb, const uchar *messages, int count, const char *file, int line)
- int _usbrecv (usbdevice *kb, const uchar *out_msg, uchar *in_msg, const char *file, int line)
- int os_usbsend (usbdevice *kb, const uchar *out_msg, int is_recv, const char *file, int line)
- int os_usbrecv (usbdevice *kb, uchar *in_msg, const char *file, int line)
- void os_sendindicators (usbdevice *kb)
- int _nk95cmd (usbdevice *kb, uchar bRequest, ushort wValue, const char *file, int line)
- int usb_tryreset (usbdevice *kb)

8.42.1 Macro Definition Documentation

8.42.1.1 #define DELAY_LONG(*kb*) usleep(100000)

Definition at line 100 of file usb.h.

Referenced by _resetusb(), _setupusb(), _usbrecv(), _usbsend(), cmd_hwload_kb(), cmd_hwload_mouse(), cmd_hwsave_kb(), and cmd_hwsave_mouse().

8.42.1.2 #define DELAY_MEDIUM(*kb*) usleep((int)(kb)->usbdelay * 10000)

Definition at line 99 of file usb.h.

Referenced by _usbrecv(), and setactive_kb().

8.42.1.3 #define DELAY_SHORT(*kb*) usleep((int)(kb)->usbdelay * 1000)

Definition at line 98 of file usb.h.

Referenced by _usbrecv(), _usbsend(), and updateindicators_kb().

8.42.1.4 #define IS_FULLRANGE(*kb*) (IS_RGB((kb)->vendor, (kb)->product) && (kb)->product != P_K65 && (kb)->product != P_K70 && (kb)->product != P_K95)

Definition at line 91 of file usb.h.

Referenced by readcmd(), and updatergb_kb().

8.42.1.5 #define IS_K65(*kb*) ((kb)->vendor == V_CORSAIR && ((kb)->product == P_K65 || (kb)->product == P_K65_NRGB || (kb)->product == P_K65_LUX || (kb)->product == P_K65_RFIRE))

Definition at line 19 of file usb.h.

Referenced by has_key().

8.42.1.6 #define IS_K70(*kb*) ((kb)->vendor == V_CORSAIR && ((kb)->product == P_K70 || (kb)->product == P_K70_NRGB || (kb)->product == P_K70_RFIRE || (kb)->product == P_K70_RFIRE_NRGB || (kb)->product == P_K70_LUX || (kb)->product == P_K70_LUX_NRGB))

Definition at line 33 of file usb.h.

8.42.1.7 #define IS_K95(*kb*) ((kb)->vendor == V_CORSAIR && ((kb)->product == P_K95 || (kb)->product == P_K95_NRGB || (kb)->product == P_K95_PLATINUM))

Definition at line 41 of file usb.h.

Referenced by cmd_hwload_kb(), cmd_hwsave_kb(), and has_key().

8.42.1.8 #define IS_M65(*kb*) ((kb)->vendor == V_CORSAIR && ((kb)->product == P_M65 || (kb)->product == P_M65_PRO))

Definition at line 53 of file usb.h.

Referenced by isblack().

8.42.1.9 #define IS_MONOCHROME(*vendor, product*) ((vendor) == (V_CORSAIR) && (product) == (P_STRAFE_NRGB))

Definition at line 86 of file usb.h.

Referenced by _setupusb().

8.42.1.10 #define IS_MONOCHROME_DEV(*kb*) IS_MONOCHROME((kb)->vendor, (kb)->product)

Definition at line 88 of file usb.h.

8.42.1.11 #define IS_MOUSE(*vendor, product*) ((vendor) == (V_CORSAIR) && ((product) == (P_M65) || (product) == (P_M65_PRO) || (product) == (P_SABRE_O) || (product) == (P_SABRE_L) || (product) == (P_SABRE_N) || (product) == (P_SCIMITAR) || (product) == (P_SCIMITAR_PRO) || (product) == (P_SABRE_O2)))

Definition at line 94 of file usb.h.

Referenced by _setupusb(), get_vtable(), has_key(), and os_inputmain().

8.42.1.12 #define IS_MOUSE_DEV(*kb*) IS_MOUSE((kb)->vendor, (kb)->product)

Definition at line 95 of file usb.h.

Referenced by readcmd().

8.42.1.13 #define IS_RGB(*vendor, product*) ((vendor) == (V_CORSAIR) && (product) != (P_K65_NRGB) && (product) != (P_K70_NRGB) && (product) != (P_K95_NRGB))

Definition at line 85 of file usb.h.

Referenced by _setupusb(), get_vtable(), and os_inputmain().

8.42.1.14 #define IS_RGB_DEV(*kb*) IS_RGB((kb)->vendor, (kb)->product)

Definition at line 87 of file usb.h.

8.42.1.15 #define IS_SABRE(*kb*) ((kb)->vendor == V_CORSAIR && ((kb)->product == P_SABRE_O || (kb)->product == P_SABRE_L || (kb)->product == P_SABRE_N || (kb)->product == P_SABRE_O2))

Definition at line 63 of file usb.h.

Referenced by has_key(), loadrgb_mouse(), and savergb_mouse().

8.42.1.16 #define IS_SCIMITAR(*kb*) ((kb)->vendor == V_CORSAIR && ((kb)->product == P_SCIMITAR || (kb)->product == P_SCIMITAR_PRO))

Definition at line 69 of file usb.h.

Referenced by has_key(), loadrgb_mouse(), and savergb_mouse().

8.42.1.17 #define IS_STRAFE(*kb*) ((kb)->vendor == V_CORSAIR && ((kb)->product == P_STRAFE || (kb)->product == P_STRAFE_NRGB))

Definition at line 47 of file usb.h.

Referenced by savergb_kb().

8.42.1.18 #define NK95_HWOFF 0x020030

Definition at line 146 of file usb.h.

Referenced by start_kb_nrgb().

8.42.1.19 #define NK95_HWON 0x020001

Definition at line 147 of file usb.h.

Referenced by revertusb().

8.42.1.20 #define NK95_M1 0x140001

Definition at line 148 of file usb.h.

Referenced by setmodeindex_nrgb().

8.42.1.21 #define NK95_M2 0x140002

Definition at line 149 of file usb.h.

Referenced by setmodeindex_nrgb().

8.42.1.22 #define NK95_M3 0x140003

Definition at line 150 of file usb.h.

Referenced by setmodeindex_nrgb().

8.42.1.23 #define nk95cmd(*kb, command*) _nk95cmd(kb, (command) >> 16 & 0xFF, (command) & 0xFFFF, __FILE_NOPATH__, __LINE__)

Definition at line 144 of file usb.h.

Referenced by revertusb(), setmodeindex_nrgb(), and start_kb_nrgb().

8.42.1.24 #define P_K65 0x1b17

Definition at line 11 of file usb.h.

Referenced by product_str().

8.42.1.25 #define P_K65_LUX 0x1b37

Definition at line 15 of file usb.h.

Referenced by product_str().

8.42.1.26 #define P_K65_LUX_STR "1b37"

Definition at line 16 of file usb.h.

8.42.1.27 #define P_K65_NRGB 0x1b07

Definition at line 13 of file usb.h.

Referenced by product_str().

8.42.1.28 #define P_K65_NRGB_STR "1b07"

Definition at line 14 of file usb.h.

8.42.1.29 #define P_K65_RFIRE 0x1b39

Definition at line 17 of file usb.h.

Referenced by product_str().

8.42.1.30 #define P_K65_RFIRE_STR "1b39"

Definition at line 18 of file usb.h.

8.42.1.31 #define P_K65_STR "1b17"

Definition at line 12 of file usb.h.

8.42.1.32 #define P_K70 0x1b13

Definition at line 21 of file usb.h.

Referenced by product_str().

8.42.1.33 #define P_K70_LUX 0x1b33

Definition at line 25 of file usb.h.

Referenced by product_str().

8.42.1.34 #define P_K70_LUX_NRGB 0x1b36

Definition at line 27 of file usb.h.

Referenced by product_str().

8.42.1.35 #define P_K70_LUX_NRGB_STR "1b36"

Definition at line 28 of file usb.h.

8.42.1.36 #define P_K70_LUX_STR "1b33"

Definition at line 26 of file usb.h.

8.42.1.37 #define P_K70_NRGB 0x1b09

Definition at line 23 of file usb.h.

Referenced by product_str().

8.42.1.38 #define P_K70_NRGB_STR "1b09"

Definition at line 24 of file usb.h.

8.42.1.39 #define P_K70_RFIRE 0x1b38

Definition at line 29 of file usb.h.

Referenced by product_str().

8.42.1.40 #define P_K70_RFIRE_NRGB 0x1b3a

Definition at line 31 of file usb.h.

Referenced by product_str().

8.42.1.41 #define P_K70_RFIRE_NRGB_STR "1b3a"

Definition at line 32 of file usb.h.

8.42.1.42 #define P_K70_RFIRE_STR "1b38"

Definition at line 30 of file usb.h.

8.42.1.43 #define P_K70_STR "1b13"

Definition at line 22 of file usb.h.

8.42.1.44 #define P_K95 0x1b11

Definition at line 35 of file usb.h.

Referenced by product_str().

8.42.1.45 #define P_K95_NRGB 0x1b08

Definition at line 37 of file usb.h.

Referenced by _nk95cmd(), and product_str().

8.42.1.46 #define P_K95_NRGB_STR "1b08"

Definition at line 38 of file usb.h.

8.42.1.47 #define P_K95_PLATINUM 0x1b2d

Definition at line 39 of file usb.h.

Referenced by product_str().

8.42.1.48 #define P_K95_PLATINUM_STR "1b2d"

Definition at line 40 of file usb.h.

8.42.1.49 #define P_K95_STR "1b11"

Definition at line 36 of file usb.h.

8.42.1.50 #define P_M65 0x1b12

Definition at line 49 of file usb.h.

Referenced by product_str().

8.42.1.51 #define P_M65_PRO 0x1b2e

Definition at line 51 of file usb.h.

Referenced by product_str().

8.42.1.52 #define P_M65_PRO_STR "1b2e"

Definition at line 52 of file usb.h.

8.42.1.53 #define P_M65_STR "1b12"

Definition at line 50 of file usb.h.

8.42.1.54 #define P_SABRE_L 0x1b19 /* laser */

Definition at line 57 of file usb.h.

Referenced by product_str().

8.42.1.55 #define P_SABRE_L_STR "1b19"

Definition at line 58 of file usb.h.

8.42.1.56 #define P_SABRE_N 0x1b2f /* new? */

Definition at line 59 of file usb.h.

Referenced by product_str().

8.42.1.57 #define P_SABRE_N_STR "1b2f"

Definition at line 60 of file usb.h.

8.42.1.58 #define P_SABRE_O 0x1b14 /* optical */

Definition at line 55 of file usb.h.

Referenced by product_str().

8.42.1.59 #define P_SABRE_O2 0x1b32 /* Observed on a CH-9000111-EU model SABRE */

Definition at line 61 of file usb.h.

Referenced by product_str().

8.42.1.60 #define P_SABRE_O2_STR "1b32"

Definition at line 62 of file usb.h.

8.42.1.61 #define P_SABRE_O_STR "1b14"

Definition at line 56 of file usb.h.

8.42.1.62 #define P_SCIMITAR 0x1b1e

Definition at line 65 of file usb.h.

Referenced by product_str().

8.42.1.63 #define P_SCIMITAR_PRO 0x1b3e

Definition at line 67 of file usb.h.

Referenced by product_str().

8.42.1.64 #define P_SCIMITAR_PRO_STR "1b3e"

Definition at line 68 of file usb.h.

8.42.1.65 #define P_SCIMITAR_STR "1b1e"

Definition at line 66 of file usb.h.

8.42.1.66 #define P_STRAFE 0x1b20

Definition at line 43 of file usb.h.

Referenced by product_str().

8.42.1.67 #define P_STRAFE_NRGB 0x1b15

Definition at line 45 of file usb.h.

Referenced by product_str().

8.42.1.68 #define P_STRAFE_NRGB_STR "1b15"

Definition at line 46 of file usb.h.

8.42.1.69 #define P_STRAFE_STR "1b20"

Definition at line 44 of file usb.h.

8.42.1.70 #define resetusb(*kb*) _resetusb(kb, __FILE_NOPATH__, __LINE__)

Definition at line 125 of file usb.h.

Referenced by usb_tryreset().

8.42.1.71 #define USB_DELAY_DEFAULT 5

Definition at line 101 of file usb.h.

Referenced by _setupusb(), and start_dev().

8.42.1.72 #define usbrecv(*kb, out_msg, in_msg*) _usbrecv(kb, out_msg, in_msg, __FILE_NOPATH__, __LINE__)

Definition at line 133 of file usb.h.

Referenced by cmd_hwload_kb(), cmd_hwload_mouse(), getfwversion(), hwloadmode(), loaddpi(), loadrgb_kb(), and loadrgb_mouse().

8.42.1.73 #define usbsend(*kb, messages, count*) _usbsend(kb, messages, count, __FILE_NOPATH__, __LINE__)

Definition at line 130 of file usb.h.

Referenced by cmd_hwsave_kb(), cmd_hwsave_mouse(), cmd_pollrate(), fwupdate(), loadrgb_kb(), savedpi(), savergb_kb(), savergb_mouse(), setactive_kb(), setactive_mouse(), updatedpi(), updatergb_kb(), and updatergb_mouse().

8.42.1.74 #define V_CORSAIR 0x1b1c

Definition at line 8 of file usb.h.

Referenced by usb_add_device(), and vendor_str().

8.42.1.75 #define V_CORSAIR_STR "1b1c"

Definition at line 9 of file usb.h.

Referenced by udev_enum(), and usb_add_device().

8.42.2 Function Documentation

8.42.2.1 int _nk95cmd (usbdevice * *kb,* uchar *bRequest,* ushort *wValue,* const char * *file,* int *line*)

Definition at line 74 of file usb_linux.c.

References ckb_err_fn, usbdevice::handle, P_K95_NRGB, and usbdevice::product.

```
74                                                                                  {
75     if(kb->product != P_K95_NRGB)
76         return 0;
77     struct usbdevfs_ctrltransfer transfer = { 0x40, bRequest, wValue, 0, 0, 5000, 0 };
78     int res = ioctl(kb->handle - 1, USBDEVFS_CONTROL, &transfer);
79     if(res <= 0){
80         ckb_err_fn("%s\n", file, line, res ? strerror(errno) : "No data written");
81         return 1;
82     }
83     return 0;
84 }
```

8.42.2.2 int _resetusb (usbdevice * *kb,* const char * *file,* int *line*)

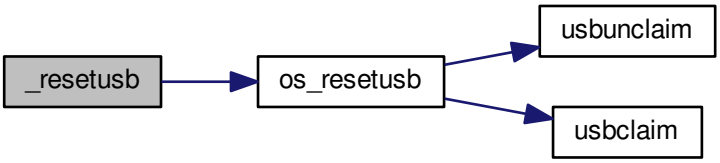
Definition at line 149 of file usb.c.

References usbdevice::active, DELAY_LONG, os_resetusb(), and usbdevice::vtable.

```
149                                                  {
150     // Perform a USB reset
151     DELAY_LONG(kb);
152     int res = os_resetusb(kb, file, line);
153     if(res)
154         return res;
155     DELAY_LONG(kb);
156     // Re-initialize the device
157     if(kb->vtable->start(kb, kb->active) != 0)
158         return -1;
```

```
159     if(kb->vtable->updatergb(kb, 1) != 0)
160         return -1;
161     return 0;
162 }
```

Here is the call graph for this function:



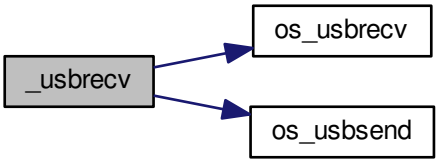
8.42.2.3 int _usbrecv (usbdevice * *kb*, const uchar * *out_msg*, uchar * *in_msg*, const char * *file*, int *line*)

Definition at line 207 of file usb.c.

References ckb_err_fn, DELAY_LONG, DELAY_MEDIUM, DELAY_SHORT, hwload_mode, os_usbrecv(), os_usbsend(), and reset_stop.

```
207                                                                                                 {
208     // Try a maximum of 3 times
209     for(int try = 0; try < 5; try++){
210         // Send the output message
211         DELAY_SHORT(kb);
212         int res = os_usbsend(kb, out_msg, 1, file, line);
213         if(res == 0)
214             return 0;
215         else if(res == -1){
216             // Retry on temporary failure
217             if(reset_stop)
218                 return 0;
219             DELAY_LONG(kb);
220             continue;
221         }
222         // Wait for the response
223         DELAY_MEDIUM(kb);
224         res = os_usbrecv(kb, in_msg, file, line);
225         if(res == 0)
226             return 0;
227         else if(res != -1)
228             return res;
229         if(reset_stop || hwload_mode != 2)
230             return 0;
231         DELAY_LONG(kb);
232     }
233     // Give up
234     ckb_err_fn("Too many send/recv failures. Dropping.\n", file, line);
235     return 0;
236 }
```

Here is the call graph for this function:



8.42.2.4 int _usbsend (usbdevice * *kb*, const uchar * *messages*, int *count*, const char * *file*, int *line*)

Definition at line 184 of file usb.c.

References DELAY_LONG, DELAY_SHORT, hwload_mode, MSG_SIZE, os_usbsend(), and reset_stop.

```
184                                                                                                 {
185     int total_sent = 0;
186     for(int i = 0; i < count; i++){
187         // Send each message via the OS function
188         while(1){
189             DELAY_SHORT(kb);
190             int res = os_usbsend(kb, messages + i * MSG_SIZE, 0, file, line);
191             if(res == 0)
192                 return 0;
193             else if(res != -1){
194                 total_sent += res;
195                 break;
196             }
197             // Stop immediately if the program is shutting down or hardware load is set to tryonce
198             if(reset_stop || hwload_mode != 2)
199                 return 0;
200             // Retry as long as the result is temporary failure
201             DELAY_LONG(kb);
202         }
203     }
204     return total_sent;
205 }
```

Here is the call graph for this function:

8.42.2.5 int closeusb (usbdevice * *kb*)

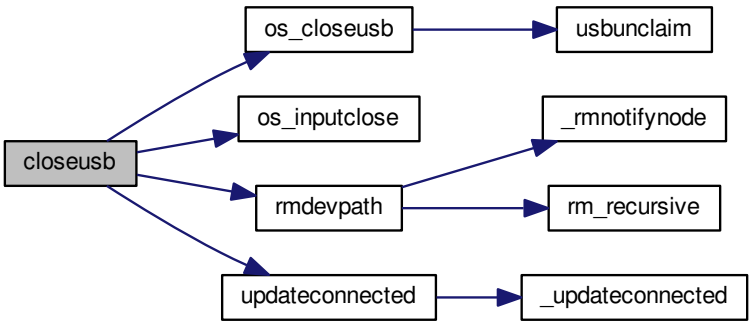
Definition at line 238 of file usb.c.

References ckb_info, devpath, dmutex, usbdevice::handle, imutex, INDEX_OF, keyboard, os_closeusb(), os_inputclose(), rmdevpath(), usbdevice::thread, updateconnected(), and usbdevice::vtable.

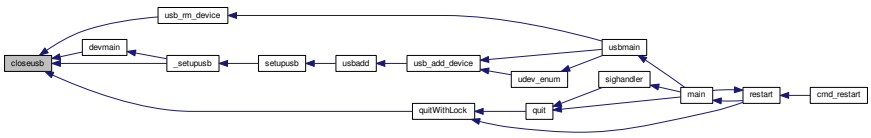
Referenced by _setupusb(), devmain(), quitWithLock(), and usb_rm_device().

```
238                             {
239     pthread_mutex_lock(imutex(kb));
240     if(kb->handle){
241         int index = INDEX_OF(kb, keyboard);
242         ckb_info("Disconnecting %s%d\n", devpath, index);
243         os_inputclose(kb);
244         updateconnected();
245         // Close USB device
246         os_closeusb(kb);
247     } else
248         updateconnected();
249     rmdevpath(kb);
250
251     // Wait for thread to close
252     pthread_mutex_unlock(imutex(kb));
253     pthread_mutex_unlock(dmutex(kb));
254     pthread_join(kb->thread, 0);
255     pthread_mutex_lock(dmutex(kb));
256
257     // Delete the profile and the control path
258     if(!kb->vtable)
259         return 0;
260     kb->vtable->freeprofile(kb);
261     memset(kb, 0, sizeof(usbdevice));
262     return 0;
263 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



8.42.2.6 void os_closeusb (usbdevice * *kb*)

Definition at line 214 of file usb_linux.c.

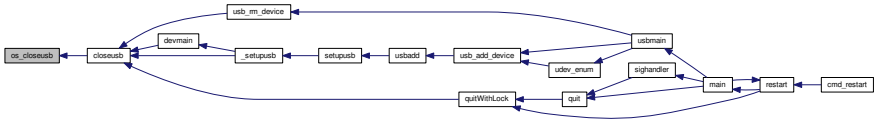
References usbdevice::handle, INDEX_OF, kbsyspath, keyboard, usbdevice::udev, and usbunclaim().

Referenced by closeusb().

```
214                                 {
215     if(kb->handle){
216         usbunclaim(kb, 0);
217         close(kb->handle - 1);
218     }
219     if(kb->udev)
220         udev_device_unref(kb->udev);
221     kb->handle = 0;
222     kb->udev = 0;
223     kbsyspath[INDEX_OF(kb, keyboard)][0] = 0;
224 }
```

Here is the call graph for this function:

Here is the caller graph for this function:



8.42.2.7 void* os_inputmain (void * *kb*)

Definition at line 93 of file usb_linux.c.

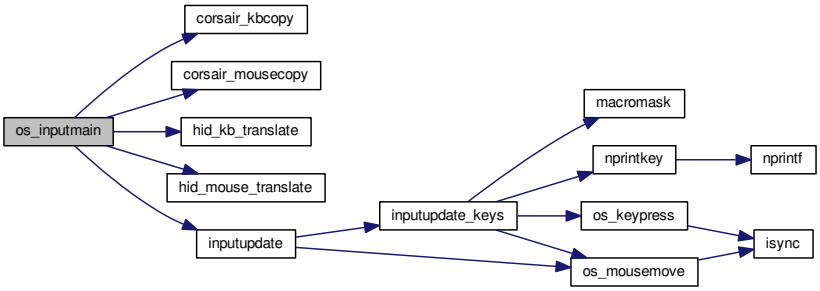
References usbdevice::active, ckb_info, corsair_kbcopy(), corsair_mousecopy(), devpath, usbdevice::epcount, usbdevice::handle, hid_kb_translate(), hid_mouse_translate(), imutex, INDEX_OF, usbdevice::input, inputupdate(), IS_MOUSE, IS_RGB, keyboard, usbinput::keys, MSG_SIZE, usbdevice::product, usbinput::rel_x, usbinput::rel_y, and usbdevice::vendor.

Referenced by _setupusb().

```
93                                  {
94     usbdevice* kb = context;
95     int fd = kb->handle - 1;
96     short vendor = kb->vendor, product = kb->product;
97     int index = INDEX_OF(kb, keyboard);
98     ckb_info("Starting input thread for %s%d\n", devpath, index);
99
100     // Monitor input transfers on all endpoints for non-RGB devices
101     // For RGB, monitor all but the last, as it's used for input/output
102     int urbcount = IS_RGB(vendor, product) ? (kb->epcount - 1) : kb->
      epcount;
103     struct usbdevfs_urb urbs[urbcount];
104     memset(urbs, 0, sizeof(urbs));
105     urbs[0].buffer_length = 8;
106     if(IS_RGB(vendor, product)){
107         if(IS_MOUSE(vendor, product))
108             urbs[1].buffer_length = 10;
109         else
110             urbs[1].buffer_length = 21;
111         urbs[2].buffer_length = MSG_SIZE;
112         if(urbcount != 3)
113             urbs[urbcount - 1].buffer_length = MSG_SIZE;
114     } else {
```

```
115         urbs[1].buffer_length = 4;
116         urbs[2].buffer_length = 15;
117     }
118     // Submit URBs
119     for(int i = 0; i < urbcount; i++){
120         urbs[i].type = USBDEVFS_URB_TYPE_INTERRUPT;
121         urbs[i].endpoint = 0x80 | (i + 1);
122         urbs[i].buffer = malloc(urbs[i].buffer_length);
123         ioctl(fd, USBDEVFS_SUBMITURB, urbs + i);
124     }
125     // Start monitoring input
126     while(1){
127         struct usbdevfs_urb* urb = 0;
128         if(ioctl(fd, USBDEVFS_REAPURB, &urb)){
129             if(errno == ENODEV || errno == ENOENT || errno == ESHUTDOWN)
130                 // Stop the thread if the handle closes
131                 break;
132             else if(errno == EPIPE && urb){
133                 // On EPIPE, clear halt on the endpoint
134                 ioctl(fd, USBDEVFS_CLEAR_HALT, &urb->endpoint);
135                 // Re-submit the URB
136                 if(urb)
137                     ioctl(fd, USBDEVFS_SUBMITURB, urb);
138                 urb = 0;
139             }
140         }
141         if(urb){
142             // Process input (if any)
143             pthread_mutex_lock(imutex(kb));
144             if(IS_MOUSE(vendor, product)){
145                 switch(urb->actual_length){
146                 case 8:
147                 case 10:
148                 case 11:
149                     // HID mouse input
150                     hid_mouse_translate(kb->input.keys, &kb->
    input.rel_x, &kb->input.rel_y, -(urb->endpoint & 0xF), urb->actual_length, urb->buffer)
    ;
151                     break;
152                 case MSG_SIZE:
153                     // Corsair mouse input
154                     corsair_mousecopy(kb->input.keys, -(urb->endpoint & 0xF), urb
    ->buffer);
155                     break;
156                 }
157             } else if(IS_RGB(vendor, product)){
158                 switch(urb->actual_length){
159                 case 8:
160                     // RGB EP 1: 6KRO (BIOS mode) input
161                     hid_kb_translate(kb->input.keys, -1, urb->actual_length, urb->
    buffer);
162                     break;
163                 case 21:
164                 case 5:
165                     // RGB EP 2: NKRO (non-BIOS) input. Accept only if keyboard is inactive
166                     if(!kb->active)
167                         hid_kb_translate(kb->input.keys, -2, urb->actual_length,
    urb->buffer);
168                     break;
169                 case MSG_SIZE:
170                     // RGB EP 3: Corsair input
171                     corsair_kbcopy(kb->input.keys, -(urb->endpoint & 0xF), urb->
    buffer);
172                     break;
173                 }
174             } else
175                 // Non-RGB input
176                 hid_kb_translate(kb->input.keys, urb->endpoint & 0xF, urb->
    actual_length, urb->buffer);
177             inputupdate(kb);
178             pthread_mutex_unlock(imutex(kb));
179             // Re-submit the URB
180             ioctl(fd, USBDEVFS_SUBMITURB, urb);
181             urb = 0;
182         }
183     }
184     // Clean up
185     ckb_info("Stopping input thread for %s%d\n", devpath, index);
186     for(int i = 0; i < urbcount; i++){
187         ioctl(fd, USBDEVFS_DISCARDURB, urbs + i);
188         free(urbs[i].buffer);
189     }
190     return 0;
191 }
```

Here is the call graph for this function:



Here is the caller graph for this function:

8.42.2.8 int os_resetusb (usbdevice * *kb,* const char * *file,* int *line*)

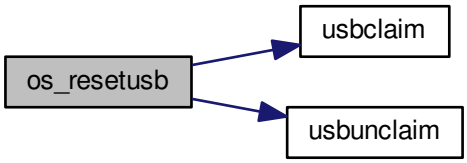
Definition at line 245 of file usb_linux.c.

References usbdevice::handle, TEST_RESET, usbclaim(), and usbunclaim().

Referenced by _resetusb().

```
245                                                          {
246     TEST_RESET(usbunclaim(kb, 1));
247     TEST_RESET(ioctl(kb->handle - 1, USBDEVFS_RESET));
248     TEST_RESET(usbclaim(kb));
249     // Success!
250     return 0;
251 }
```

Here is the call graph for this function:



Here is the caller graph for this function:

8.42.2.9 void os_sendindicators (usbdevice * kb)

Definition at line 86 of file usb_linux.c.

References ckb_err, usbdevice::handle, and usbdevice::ileds.

Referenced by updateindicators_kb().

```
86                                      {
87      struct usbdevfs_ctrltransfer transfer = { 0x21, 0x09, 0x0200, 0x00, 1, 500, &kb->
       ileds };
88      int res = ioctl(kb->handle - 1, USBDEVFS_CONTROL, &transfer);
89      if(res <= 0)
90          ckb_err("%s\n", res ? strerror(errno) : "No data written");
91  }
```

Here is the caller graph for this function:

8.42.2.10 int os_setupusb (usbdevice * kb)

Definition at line 271 of file usb_linux.c.

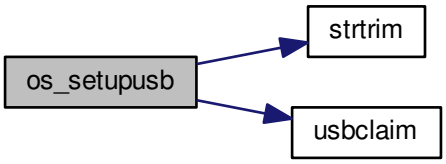
References ckb_err, ckb_info, ckb_warn, devpath, usbdevice::epcount, FEAT_RGB, usbdevice::fwversion, HAS_-FEATURES, INDEX_OF, KB_NAME_LEN, keyboard, usbdevice::name, usbdevice::serial, SERIAL_LEN, strtrim(), usbdevice::udev, and usbclaim().

Referenced by _setupusb().

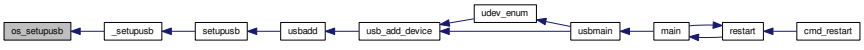
```
271                                 {
272     // Copy device description and serial
273     struct udev_device* dev = kb->udev;
274     const char* name = udev_device_get_sysattr_value(dev, "product");
275     if(name)
276         strncpy(kb->name, name, KB_NAME_LEN);
277     strtrim(kb->name);
278     const char* serial = udev_device_get_sysattr_value(dev, "serial");
279     if(serial)
280         strncpy(kb->serial, serial, SERIAL_LEN);
281     strtrim(kb->serial);
282     // Copy firmware version (needed to determine USB protocol)
```

```
283     const char* firmware = udev_device_get_sysattr_value(dev, "bcdDevice");
284     if(firmware)
285         sscanf(firmware, "%hx", &kb->fwversion);
286     else
287         kb->fwversion = 0;
288     int index = INDEX_OF(kb, keyboard);
289     ckb_info("Connecting %s at %s%d\n", kb->name, devpath, index);
290
291     // Claim the USB interfaces
292     const char* ep_str = udev_device_get_sysattr_value(dev, "bNumInterfaces");
293     kb->epcount = 0;
294     if(ep_str)
295         sscanf(ep_str, "%d", &kb->epcount);
296     if(kb->epcount == 0){
297         // This shouldn't happen, but if it does, assume EP count based on what the device is supposed to
      have
298         kb->epcount = (HAS_FEATURES(kb, FEAT_RGB) ? 4 : 3);
299         ckb_warn("Unable to read endpoint count from udev, assuming %d...\n", kb->
     epcount);
300     }
301     if(usbclaim(kb)){
302         ckb_err("Failed to claim interfaces: %s\n", strerror(errno));
303         return -1;
304     }
305     return 0;
306 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



8.42.2.11 int os_usbrecv (usbdevice * *kb*, uchar * *in_msg*, const char * *file*, int *line*)

Definition at line 42 of file usb_linux.c.

References ckb_err_fn, ckb_warn_fn, usbdevice::epcount, usbdevice::handle, and MSG_SIZE.

Referenced by _usbrecv().

```
42                                                                          {
43     int res;
44     // This is what CUE does, but it doesn't seem to work on linux.
45     /*if(kb->fwversion >= 0x130){
46         struct usbdevfs_bulktransfer transfer;
47         memset(&transfer, 0, sizeof(transfer));
48         transfer.ep = 0x84;
49         transfer.len = MSG_SIZE;
50         transfer.timeout = 5000;
51         transfer.data = in_msg;
52         res = ioctl(kb->handle - 1, USBDEVFS_BULK, &transfer);
```

```
53     } else {*/
54         struct usbdevfs_ctrltransfer transfer = { 0xa1, 0x01, 0x0300, kb->
       epcount - 1, MSG_SIZE, 5000, in_msg };
55         res = ioctl(kb->handle - 1, USBDEVFS_CONTROL, &transfer);
56     //}
57     if(res <= 0){
58         ckb_err_fn("%s\n", file, line, res ? strerror(errno) : "No data read");
59         if(res == -1 && errno == ETIMEDOUT)
60             return -1;
61         else
62             return 0;
63     } else if(res != MSG_SIZE)
64         ckb_warn_fn("Read %d bytes (expected %d)\n", file, line, res,
       MSG_SIZE);
65 #ifdef DEBUG_USB_RECV
66     char converted[MSG_SIZE*3 + 1];
67     for(int i=0;i<MSG_SIZE;i++)
68         sprintf(&converted[i*3], "%02x ", in_msg[i]);
69     ckb_warn_fn("Recv %s\n", file, line, converted);
70 #endif
71     return res;
72 }
```

Here is the caller graph for this function:

8.42.2.12 int os_usbsend (usbdevice * *kb,* const uchar * *out_msg,* int *is_recv,* const char * *file,* int *line*)

Definition at line 11 of file usb_linux.c.

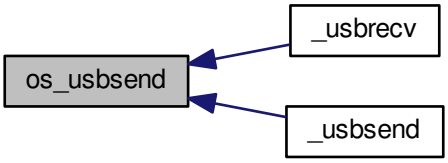
References ckb_err_fn, ckb_warn_fn, usbdevice::epcount, usbdevice::fwversion, usbdevice::handle, and MSG_SIZE.

Referenced by _usbrecv(), and _usbsend().

```
11                                                                                                                  {
12     int res;
13     if(kb->fwversion >= 0x120 && !is_recv){
14         struct usbdevfs_bulktransfer transfer;
15         memset(&transfer, 0, sizeof(transfer));
16         transfer.ep = (kb->fwversion >= 0x130 && kb->fwversion < 0x200) ? 4 : 3;
17         transfer.len = MSG_SIZE;
18         transfer.timeout = 5000;
19         transfer.data = (void*)out_msg;
20         res = ioctl(kb->handle - 1, USBDEVFS_BULK, &transfer);
21     } else {
22         struct usbdevfs_ctrltransfer transfer = { 0x21, 0x09, 0x0200, kb->
       epcount - 1, MSG_SIZE, 5000, (void*)out_msg };
23         res = ioctl(kb->handle - 1, USBDEVFS_CONTROL, &transfer);
24     }
25     if(res <= 0){
26         ckb_err_fn("%s\n", file, line, res ? strerror(errno) : "No data written");
27         if(res == -1 && errno == ETIMEDOUT)
28             return -1;
29         else
30             return 0;
31     } else if(res != MSG_SIZE)
32         ckb_warn_fn("Wrote %d bytes (expected %d)\n", file, line, res,
       MSG_SIZE);
33 #ifdef DEBUG_USB
34     char converted[MSG_SIZE*3 + 1];
35     for(int i=0;i<MSG_SIZE;i++)
36         sprintf(&converted[i*3], "%02x ", out_msg[i]);
37     ckb_warn_fn("Sent %s\n", file, line, converted);
38 #endif
```

```
39     return res;
40 }
```

Here is the caller graph for this function:



8.42.2.13 const char* product_str (short *product*)

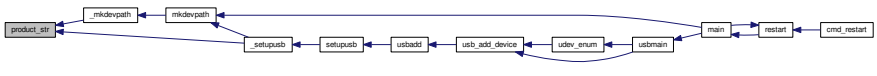
Definition at line 26 of file usb.c.

References P_K65, P_K65_LUX, P_K65_NRGB, P_K65_RFIRE, P_K70, P_K70_LUX, P_K70_LUX_NRGB, P_K70_NRGB, P_K70_RFIRE, P_K70_RFIRE_NRGB, P_K95, P_K95_NRGB, P_K95_PLATINUM, P_M65, P_M65_PRO, P_SABRE_L, P_SABRE_N, P_SABRE_O, P_SABRE_O2, P_SCIMITAR, P_SCIMITAR_PRO, P_STRAFE, and P_STRAFE_NRGB.

Referenced by _mkdevpath(), and _setupusb().

```
26                                      {
27     if(product == P_K95 || product == P_K95_NRGB || product ==
       P_K95_PLATINUM)
28         return "k95";
29     if(product == P_K70 || product == P_K70_NRGB || product ==
       P_K70_LUX || product == P_K70_LUX_NRGB || product ==
       P_K70_RFIRE || product == P_K70_RFIRE_NRGB)
30         return "k70";
31     if(product == P_K65 || product == P_K65_NRGB || product ==
       P_K65_LUX || product == P_K65_RFIRE)
32         return "k65";
33     if(product == P_STRAFE || product == P_STRAFE_NRGB)
34         return "strafe";
35     if(product == P_M65 || product == P_M65_PRO)
36         return "m65";
37     if(product == P_SABRE_O || product == P_SABRE_L || product ==
       P_SABRE_N || product == P_SABRE_O2)
38         return "sabre";
39     if(product == P_SCIMITAR || product == P_SCIMITAR_PRO)
40         return "scimitar";
41     return "";
42 }
```

Here is the caller graph for this function:



8.42.2.14 int revertusb (usbdevice * *kb*)

Definition at line 137 of file usb.c.

References FEAT_RGB, HAS_FEATURES, NEEDS_FW_UPDATE, NK95_HWON, nk95cmd, and setactive.

Referenced by quitWithLock().

```
137                             {
138     if(NEEDS_FW_UPDATE(kb))
139         return 0;
140     if(!HAS_FEATURES(kb, FEAT_RGB)){
141         nk95cmd(kb, NK95_HWON);
142         return 0;
143     }
144     if(setactive(kb, 0))
145         return -1;
146     return 0;
147 }
```

Here is the caller graph for this function:

8.42.2.15 void setupusb (usbdevice * *kb*)

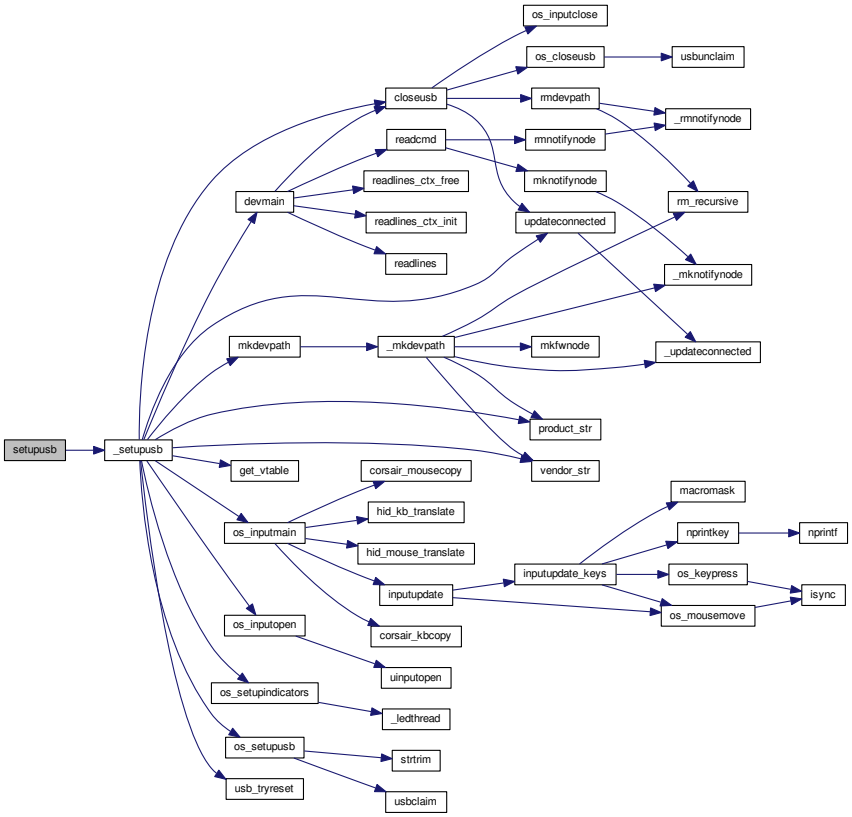
Definition at line 131 of file usb.c.

References _setupusb(), ckb_err, imutex, and usbdevice::thread.

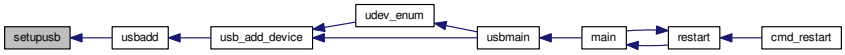
Referenced by usbadd().

```
131                             {
132     pthread_mutex_lock(imutex(kb));
133     if(pthread_create(&kb->thread, 0, _setupusb, kb))
134         ckb_err("Failed to create USB thread\n");
135 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



8.42.2.16 int usb_tryreset (usbdevice * *kb*)

Definition at line 164 of file usb.c.

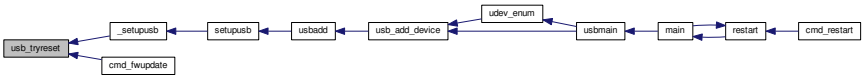
References ckb_err, ckb_info, reset_stop, and resetusb.

Referenced by _setupusb(), and cmd_fwupdate().

```
164                                 {
165     if(reset_stop)
166         return -1;
167     ckb_info("Attempting reset...\n");
168     while(1){
169         int res = resetusb(kb);
170         if(!res){
171             ckb_info("Reset success\n");
172             return 0;
173         }
174         if(res == -2 || reset_stop)
175             break;
176     }
```

```
177     ckb_err("Reset failed. Disconnecting.\n");
178     return -1;
179 }
```

Here is the caller graph for this function:



8.42.2.17 void usbkill ( )

Definition at line 492 of file usb_linux.c.

Referenced by quitWithLock().

```
492             {
493     udev_unref(udev);
494     udev = 0;
495 }
```

Here is the caller graph for this function:

8.42.2.18 int usbmain ( )

Definition at line 441 of file usb_linux.c.

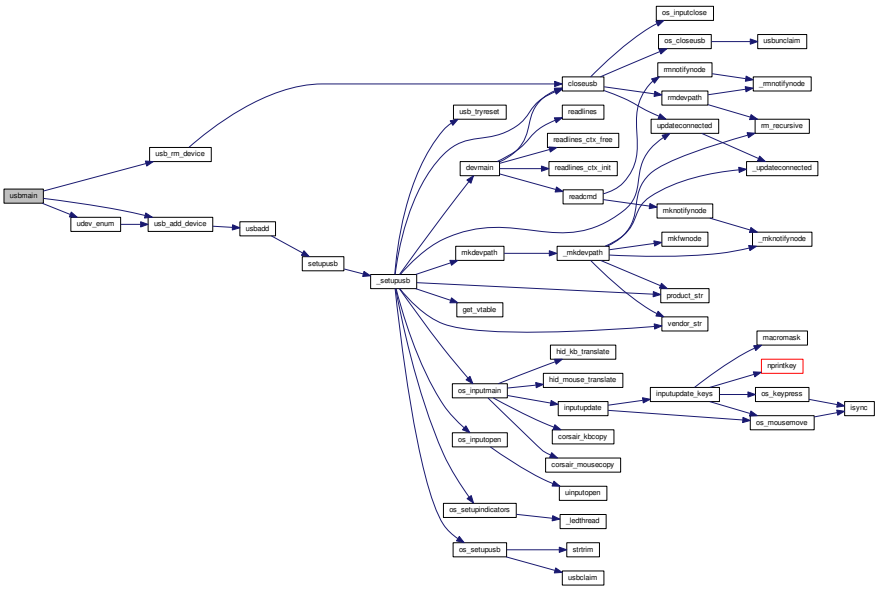
References ckb_fatal, ckb_warn, udev_enum(), usb_add_device(), and usb_rm_device().

Referenced by main().

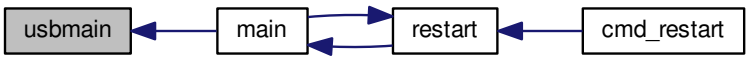
```
441             {
442     // Load the uinput module (if it's not loaded already)
443     if(system("modprobe uinput") != 0)
444         ckb_warn("Failed to load uinput module\n");
445
446     // Create the udev object
447     if(!(udev = udev_new())){
448         ckb_fatal("Failed to initialize udev\n");
449         return -1;
450     }
451
452     // Enumerate all currently connected devices
453     udev_enum();
454
455     // Done scanning. Enter a loop to poll for device updates
456     struct udev_monitor* monitor = udev_monitor_new_from_netlink(udev, "udev");
457     udev_monitor_filter_add_match_subsystem_devtype(monitor, "usb", 0);
458     udev_monitor_enable_receiving(monitor);
459     // Get an fd for the monitor
460     int fd = udev_monitor_get_fd(monitor);
461     fd_set fds;
462     while(udev){
463         FD_ZERO(&fds);
464         FD_SET(fd, &fds);
```

```
465         // Block until an event is read
466         if(select(fd + 1, &fds, 0, 0, 0) > 0 && FD_ISSET(fd, &fds)){
467             struct udev_device* dev = udev_monitor_receive_device(monitor);
468             if(!dev)
469                 continue;
470             const char* action = udev_device_get_action(dev);
471             if(!action){
472                 udev_device_unref(dev);
473                 continue;
474             }
475             // Add/remove device
476             if(!strcmp(action, "add")){
477                 int res = usb_add_device(dev);
478                 if(res == 0)
479                     continue;
480                 // If the device matched but the handle wasn't opened correctly, re-enumerate (this
    sometimes solves the problem)
481                 if(res == -1)
482                     udev_enum();
483             } else if(!strcmp(action, "remove"))
484                 usb_rm_device(dev);
485             udev_device_unref(dev);
486         }
487     }
488     udev_monitor_unref(monitor);
489     return 0;
490 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



8.42.2.19 const char* vendor_str (short *vendor*)

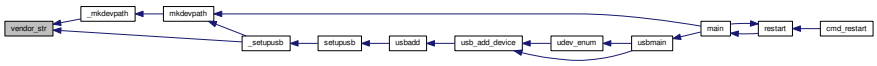
Definition at line 20 of file usb.c.

References V_CORSAIR.

Referenced by _mkdevpath(), and _setupusb().

```
20                                      {
21     if(vendor == V_CORSAIR)
22         return "corsair";
23     return "";
24 }
```

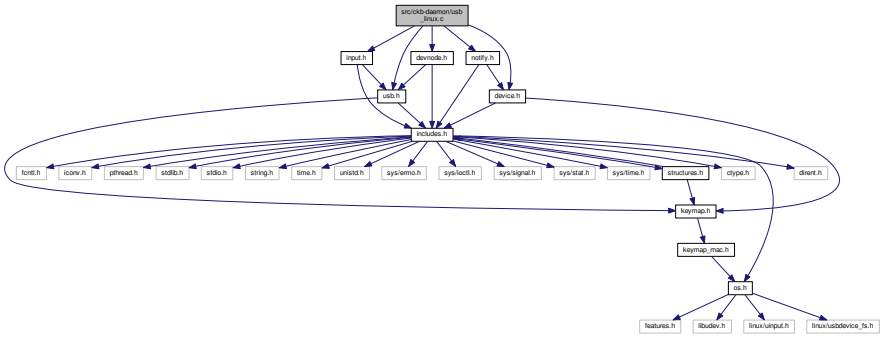
Here is the caller graph for this function:



8.43 src/ckb-daemon/usb_linux.c File Reference

```
#include "device.h"
#include "devnode.h"
#include "input.h"
#include "notify.h"
#include "usb.h"
```

Include dependency graph for usb_linux.c:



Data Structures

- struct _model

Macros

- #define TEST_RESET(op)
- #define N_MODELS (sizeof(models) / sizeof(_model))

Functions

- int os_usbsend (usbdevice *kb, const uchar *out_msg, int is_recv, const char *file, int line)

- int os_usbrecv (usbdevice *kb, uchar *in_msg, const char *file, int line)
- int _nk95cmd (usbdevice *kb, uchar bRequest, ushort wValue, const char *file, int line)
- void os_sendindicators (usbdevice *kb)
- void * os_inputmain (void *context)
- int usbunclaim (usbdevice *kb, int resetting)
- void os_closeusb (usbdevice *kb)
- int usbclaim (usbdevice *kb)
- int os_resetusb (usbdevice *kb, const char *file, int line)
- void strtrim (char *string)
- int os_setupusb (usbdevice *kb)
- int usbadd (struct udev_device *dev, short vendor, short product)
- static int usb_add_device (struct udev_device *dev)
- static void usb_rm_device (struct udev_device *dev)
- static void udev_enum ()
- int usbmain ()
- void usbkill ()

Variables

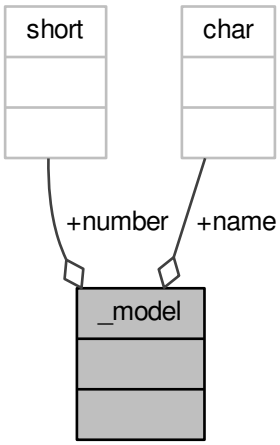
- static char kbsyspath [9][FILENAME_MAX]
- static struct udev * udev
- pthread_t usbthread
- pthread_t udevthread
- static _model models []

8.43.1 Data Structure Documentation

8.43.1.1 struct _model

Definition at line 355 of file usb_linux.c.

Collaboration diagram for _model:



Data Fields

const char *	name	
short	number	

8.43.2 Macro Definition Documentation

8.43.2.1 #define N_MODELS (sizeof(models) / sizeof(_model))

Definition at line 386 of file usb_linux.c.

Referenced by usb_add_device().

8.43.2.2 #define TEST_RESET(*op*)

Value:

```
if(op){                                                                 \
        ckb_err_fn("resetusb failed: %s\n", file, line, strerror(errno));   \
        if(errno == EINTR || errno == EAGAIN)                               \
            return -1;              /* try again if status code says so */  \
        return -2;                  /* else, remove device */               \
    }
```

Definition at line 237 of file usb_linux.c.

Referenced by os_resetusb().

8.43.3 Function Documentation

8.43.3.1 int _nk95cmd (usbdevice * *kb,* uchar *bRequest,* ushort *wValue,* const char * *file,* int *line*)

Definition at line 74 of file usb_linux.c.

References ckb_err_fn, usbdevice::handle, P_K95_NRGB, and usbdevice::product.

```
74                                                                                  {
75      if(kb->product != P_K95_NRGB)
76          return 0;
77      struct usbdevfs_ctrltransfer transfer = { 0x40, bRequest, wValue, 0, 0, 5000, 0 };
78      int res = ioctl(kb->handle - 1, USBDEVFS_CONTROL, &transfer);
79      if(res <= 0){
80          ckb_err_fn("%s\n", file, line, res ? strerror(errno) : "No data written");
81          return 1;
82      }
83      return 0;
84  }
```

8.43.3.2 void os_closeusb (usbdevice * *kb*)

Definition at line 214 of file usb_linux.c.

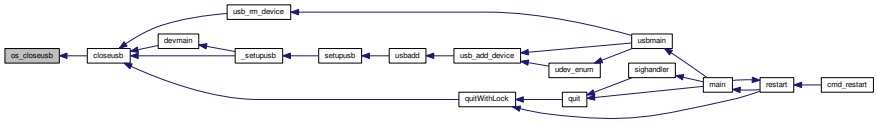
References usbdevice::handle, INDEX_OF, kbsyspath, keyboard, usbdevice::udev, and usbunclaim().

Referenced by closeusb().

```
214                                 {
215      if(kb->handle){
216          usbunclaim(kb, 0);
217          close(kb->handle - 1);
218      }
219      if(kb->udev)
220          udev_device_unref(kb->udev);
221      kb->handle = 0;
222      kb->udev = 0;
223      kbsyspath[INDEX_OF(kb, keyboard)][0] = 0;
224  }
```

Here is the call graph for this function:

Here is the caller graph for this function:



8.43.3.3 void* os_inputmain (void * *context*)

Definition at line 93 of file usb_linux.c.

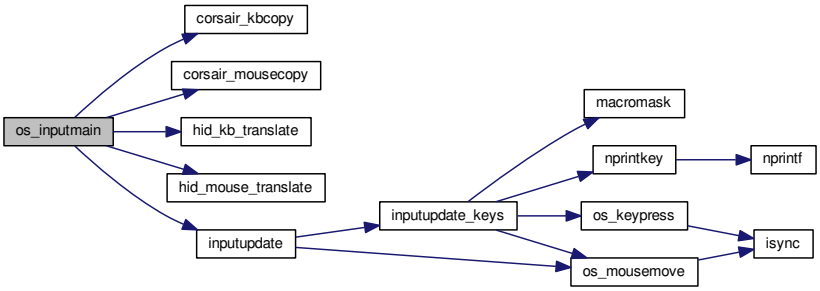
References usbdevice::active, ckb_info, corsair_kbcopy(), corsair_mousecopy(), devpath, usbdevice::epcount, usbdevice::handle, hid_kb_translate(), hid_mouse_translate(), imutex, INDEX_OF, usbdevice::input, inputupdate(), IS_MOUSE, IS_RGB, keyboard, usbinput::keys, MSG_SIZE, usbdevice::product, usbinput::rel_x, usbinput::rel_y, and usbdevice::vendor.

Referenced by _setupusb().

```
93                                {
94     usbdevice* kb = context;
95     int fd = kb->handle - 1;
96     short vendor = kb->vendor, product = kb->product;
97     int index = INDEX_OF(kb, keyboard);
98     ckb_info("Starting input thread for %s%d\n", devpath, index);
99
100     // Monitor input transfers on all endpoints for non-RGB devices
101     // For RGB, monitor all but the last, as it's used for input/output
102     int urbcount = IS_RGB(vendor, product) ? (kb->epcount - 1) : kb->
      epcount;
103     struct usbdevfs_urb urbs[urbcount];
104     memset(urbs, 0, sizeof(urbs));
105     urbs[0].buffer_length = 8;
106     if(IS_RGB(vendor, product)){
107         if(IS_MOUSE(vendor, product))
108             urbs[1].buffer_length = 10;
109         else
110             urbs[1].buffer_length = 21;
111         urbs[2].buffer_length = MSG_SIZE;
112         if(urbcount != 3)
113             urbs[urbcount - 1].buffer_length = MSG_SIZE;
114     } else {
115         urbs[1].buffer_length = 4;
116         urbs[2].buffer_length = 15;
117     }
118     // Submit URBs
119     for(int i = 0; i < urbcount; i++){
120         urbs[i].type = USBDEVFS_URB_TYPE_INTERRUPT;
121         urbs[i].endpoint = 0x80 | (i + 1);
122         urbs[i].buffer = malloc(urbs[i].buffer_length);
123         ioctl(fd, USBDEVFS_SUBMITURB, urbs + i);
124     }
125     // Start monitoring input
126     while(1){
127         struct usbdevfs_urb* urb = 0;
```

```
128         if(ioctl(fd, USBDEVFS_REAPURB, &urb)){
129             if(errno == ENODEV || errno == ENOENT || errno == ESHUTDOWN)
130                 // Stop the thread if the handle closes
131                 break;
132             else if(errno == EPIPE && urb){
133                 // On EPIPE, clear halt on the endpoint
134                 ioctl(fd, USBDEVFS_CLEAR_HALT, &urb->endpoint);
135                 // Re-submit the URB
136                 if(urb)
137                     ioctl(fd, USBDEVFS_SUBMITURB, urb);
138                 urb = 0;
139             }
140         }
141         if(urb){
142             // Process input (if any)
143             pthread_mutex_lock(imutex(kb));
144             if(IS_MOUSE(vendor, product)){
145                 switch(urb->actual_length){
146                 case 8:
147                 case 10:
148                 case 11:
149                     // HID mouse input
150                     hid_mouse_translate(kb->input.keys, &kb->
    input.rel_x, &kb->input.rel_y, -(urb->endpoint & 0xF), urb->actual_length, urb->buffer)
    ;
151                     break;
152                 case MSG_SIZE:
153                     // Corsair mouse input
154                     corsair_mousecopy(kb->input.keys, -(urb->endpoint & 0xF), urb
    ->buffer);
155                     break;
156                 }
157             } else if(IS_RGB(vendor, product)){
158                 switch(urb->actual_length){
159                 case 8:
160                     // RGB EP 1: 6KRO (BIOS mode) input
161                     hid_kb_translate(kb->input.keys, -1, urb->actual_length, urb->
    buffer);
162                     break;
163                 case 21:
164                 case 5:
165                     // RGB EP 2: NKRO (non-BIOS) input. Accept only if keyboard is inactive
166                     if(!kb->active)
167                         hid_kb_translate(kb->input.keys, -2, urb->actual_length,
    urb->buffer);
168                     break;
169                 case MSG_SIZE:
170                     // RGB EP 3: Corsair input
171                     corsair_kbcopy(kb->input.keys, -(urb->endpoint & 0xF), urb->
    buffer);
172                     break;
173                 }
174             } else
175                 // Non-RGB input
176                 hid_kb_translate(kb->input.keys, urb->endpoint & 0xF, urb->
    actual_length, urb->buffer);
177             inputupdate(kb);
178             pthread_mutex_unlock(imutex(kb));
179             // Re-submit the URB
180             ioctl(fd, USBDEVFS_SUBMITURB, urb);
181             urb = 0;
182         }
183     }
184     // Clean up
185     ckb_info("Stopping input thread for %s%d\n", devpath, index);
186     for(int i = 0; i < urbcount; i++){
187         ioctl(fd, USBDEVFS_DISCARDURB, urbs + i);
188         free(urbs[i].buffer);
189     }
190     return 0;
191 }
```

Here is the call graph for this function:



Here is the caller graph for this function:

8.43.3.4 int os_resetusb (usbdevice * *kb,* const char * *file,* int *line*)

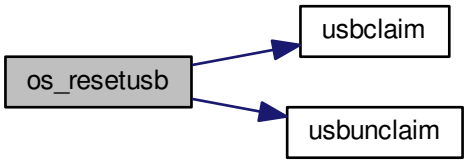
Definition at line 245 of file usb_linux.c.

References usbdevice::handle, TEST_RESET, usbclaim(), and usbunclaim().

Referenced by _resetusb().

```
245                                                         {
246     TEST_RESET(usbunclaim(kb, 1));
247     TEST_RESET(ioctl(kb->handle - 1, USBDEVFS_RESET));
248     TEST_RESET(usbclaim(kb));
249     // Success!
250     return 0;
251 }
```

Here is the call graph for this function:



Here is the caller graph for this function:

8.43.3.5 void os_sendindicators (usbdevice * kb)

Definition at line 86 of file usb_linux.c.

References ckb_err, usbdevice::handle, and usbdevice::ileds.

Referenced by updateindicators_kb().

```
86                                      {
87     struct usbdevfs_ctrltransfer transfer = { 0x21, 0x09, 0x0200, 0x00, 1, 500, &kb->
       ileds };
88     int res = ioctl(kb->handle - 1, USBDEVFS_CONTROL, &transfer);
89     if(res <= 0)
90         ckb_err("%s\n", res ? strerror(errno) : "No data written");
91 }
```

Here is the caller graph for this function:

8.43.3.6 int os_setupusb (usbdevice * kb)

Definition at line 271 of file usb_linux.c.

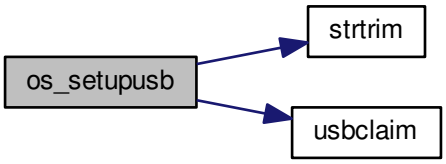
References ckb_err, ckb_info, ckb_warn, devpath, usbdevice::epcount, FEAT_RGB, usbdevice::fwversion, HAS_-FEATURES, INDEX_OF, KB_NAME_LEN, keyboard, usbdevice::name, usbdevice::serial, SERIAL_LEN, strtrim(), usbdevice::udev, and usbclaim().

Referenced by _setupusb().

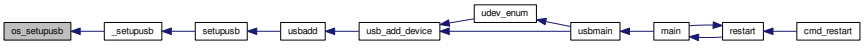
```
271                                 {
272     // Copy device description and serial
273     struct udev_device* dev = kb->udev;
274     const char* name = udev_device_get_sysattr_value(dev, "product");
275     if(name)
276         strncpy(kb->name, name, KB_NAME_LEN);
277     strtrim(kb->name);
278     const char* serial = udev_device_get_sysattr_value(dev, "serial");
279     if(serial)
280         strncpy(kb->serial, serial, SERIAL_LEN);
281     strtrim(kb->serial);
282     // Copy firmware version (needed to determine USB protocol)
```

```
283     const char* firmware = udev_device_get_sysattr_value(dev, "bcdDevice");
284     if(firmware)
285         sscanf(firmware, "%hx", &kb->fwversion);
286     else
287         kb->fwversion = 0;
288     int index = INDEX_OF(kb, keyboard);
289     ckb_info("Connecting %s at %s%d\n", kb->name, devpath, index);
290
291     // Claim the USB interfaces
292     const char* ep_str = udev_device_get_sysattr_value(dev, "bNumInterfaces");
293     kb->epcount = 0;
294     if(ep_str)
295         sscanf(ep_str, "%d", &kb->epcount);
296     if(kb->epcount == 0){
297         // This shouldn't happen, but if it does, assume EP count based on what the device is supposed to
      have
298         kb->epcount = (HAS_FEATURES(kb, FEAT_RGB) ? 4 : 3);
299         ckb_warn("Unable to read endpoint count from udev, assuming %d...\n", kb->
     epcount);
300     }
301     if(usbclaim(kb)){
302         ckb_err("Failed to claim interfaces: %s\n", strerror(errno));
303         return -1;
304     }
305     return 0;
306 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



8.43.3.7 int os_usbrecv (usbdevice * *kb*, uchar * *in_msg*, const char * *file*, int *line*)

Definition at line 42 of file usb_linux.c.

References ckb_err_fn, ckb_warn_fn, usbdevice::epcount, usbdevice::handle, and MSG_SIZE.

Referenced by _usbrecv().

```
42                                                                          {
43     int res;
44     // This is what CUE does, but it doesn't seem to work on linux.
45     /*if(kb->fwversion >= 0x130){
46         struct usbdevfs_bulktransfer transfer;
47         memset(&transfer, 0, sizeof(transfer));
48         transfer.ep = 0x84;
49         transfer.len = MSG_SIZE;
50         transfer.timeout = 5000;
51         transfer.data = in_msg;
52         res = ioctl(kb->handle - 1, USBDEVFS_BULK, &transfer);
```

```
53     } else {*/
54         struct usbdevfs_ctrltransfer transfer = { 0xa1, 0x01, 0x0300, kb->
       epcount - 1, MSG_SIZE, 5000, in_msg };
55         res = ioctl(kb->handle - 1, USBDEVFS_CONTROL, &transfer);
56     //}
57     if(res <= 0){
58         ckb_err_fn("%s\n", file, line, res ? strerror(errno) : "No data read");
59         if(res == -1 && errno == ETIMEDOUT)
60             return -1;
61         else
62             return 0;
63     } else if(res != MSG_SIZE)
64         ckb_warn_fn("Read %d bytes (expected %d)\n", file, line, res,
       MSG_SIZE);
65 #ifdef DEBUG_USB_RECV
66     char converted[MSG_SIZE*3 + 1];
67     for(int i=0;i<MSG_SIZE;i++)
68         sprintf(&converted[i*3], "%02x ", in_msg[i]);
69     ckb_warn_fn("Recv %s\n", file, line, converted);
70 #endif
71     return res;
72 }
```

Here is the caller graph for this function:

8.43.3.8 int os_usbsend (usbdevice * *kb*, const uchar * *out_msg*, int *is_recv*, const char * *file*, int *line*)

Definition at line 11 of file usb_linux.c.

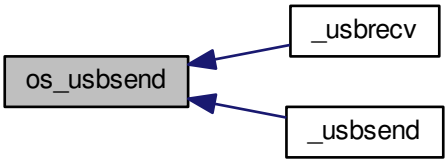
References ckb_err_fn, ckb_warn_fn, usbdevice::epcount, usbdevice::fwversion, usbdevice::handle, and MSG_SIZE.

Referenced by _usbrecv(), and _usbsend().

```
11                                                                                                                  {
12     int res;
13     if(kb->fwversion >= 0x120 && !is_recv){
14         struct usbdevfs_bulktransfer transfer;
15         memset(&transfer, 0, sizeof(transfer));
16         transfer.ep = (kb->fwversion >= 0x130 && kb->fwversion < 0x200) ? 4 : 3;
17         transfer.len = MSG_SIZE;
18         transfer.timeout = 5000;
19         transfer.data = (void*)out_msg;
20         res = ioctl(kb->handle - 1, USBDEVFS_BULK, &transfer);
21     } else {
22         struct usbdevfs_ctrltransfer transfer = { 0x21, 0x09, 0x0200, kb->
       epcount - 1, MSG_SIZE, 5000, (void*)out_msg };
23         res = ioctl(kb->handle - 1, USBDEVFS_CONTROL, &transfer);
24     }
25     if(res <= 0){
26         ckb_err_fn("%s\n", file, line, res ? strerror(errno) : "No data written");
27         if(res == -1 && errno == ETIMEDOUT)
28             return -1;
29         else
30             return 0;
31     } else if(res != MSG_SIZE)
32         ckb_warn_fn("Wrote %d bytes (expected %d)\n", file, line, res,
       MSG_SIZE);
33 #ifdef DEBUG_USB
34     char converted[MSG_SIZE*3 + 1];
35     for(int i=0;i<MSG_SIZE;i++)
36         sprintf(&converted[i*3], "%02x ", out_msg[i]);
37     ckb_warn_fn("Sent %s\n", file, line, converted);
38 #endif
```

```
39     return res;
40 }
```

Here is the caller graph for this function:



8.43.3.9 void strtrim (char * *string*)

Definition at line 253 of file usb_linux.c.

Referenced by os_setupusb().

```
253                         {
254     // Find last non-space
255     char* last = string;
256     for(char* c = string; *c != 0; c++){
257         if(!isspace(*c))
258             last = c;
259     }
260     last[1] = 0;
261     // Find first non-space
262     char* first = string;
263     for(; *first != 0; first++){
264         if(!isspace(*first))
265             break;
266     }
267     if(first != string)
268         memmove(string, first, last - first);
269 }
```

Here is the caller graph for this function:

8.43.3.10 static void udev_enum ( ) [static]

Definition at line 418 of file usb_linux.c.

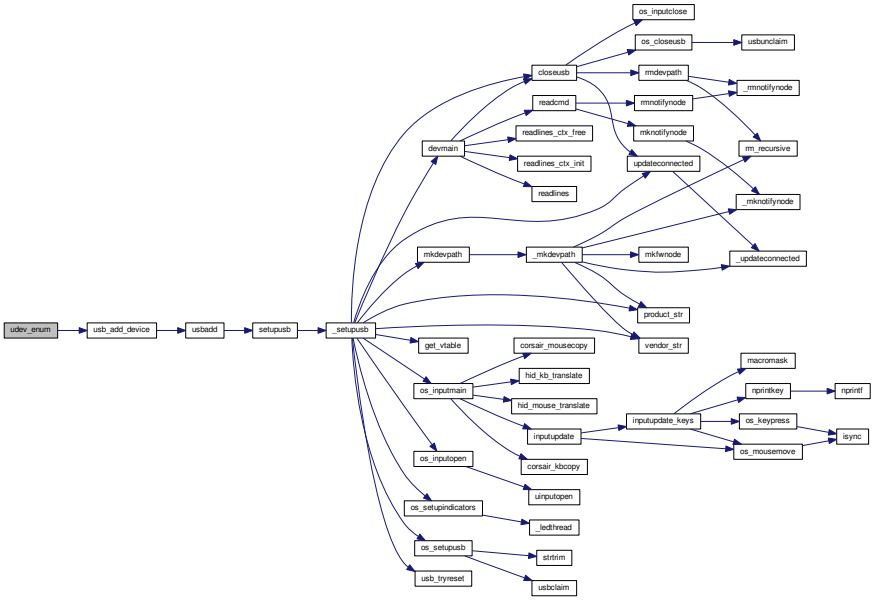
References usb_add_device(), and V_CORSAIR_STR.

Referenced by usbmain().

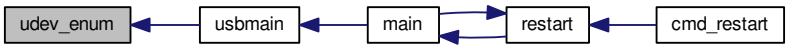
```
418                         {
419     struct udev_enumerate* enumerator = udev_enumerate_new(udev);
420     udev_enumerate_add_match_subsystem(enumerator, "usb");
421     udev_enumerate_add_match_sysattr(enumerator, "idVendor", V_CORSAIR_STR);
422     udev_enumerate_scan_devices(enumerator);
423     struct udev_list_entry* devices, *dev_list_entry;
```

```
424     devices = udev_enumerate_get_list_entry(enumerator);
425
426     udev_list_entry_foreach(dev_list_entry, devices){
427         const char* path = udev_list_entry_get_name(dev_list_entry);
428         if(!path)
429             continue;
430         struct udev_device* dev = udev_device_new_from_syspath(udev, path);
431         if(!dev)
432             continue;
433         // If the device matches a recognized device ID, open it
434         if(usb_add_device(dev))
435             // Release device if not
436             udev_device_unref(dev);
437     }
438     udev_enumerate_unref(enumerator);
439 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



8.43.3.11 static int usb_add_device (struct udev_device * *dev*) [static]

Definition at line 389 of file usb_linux.c.

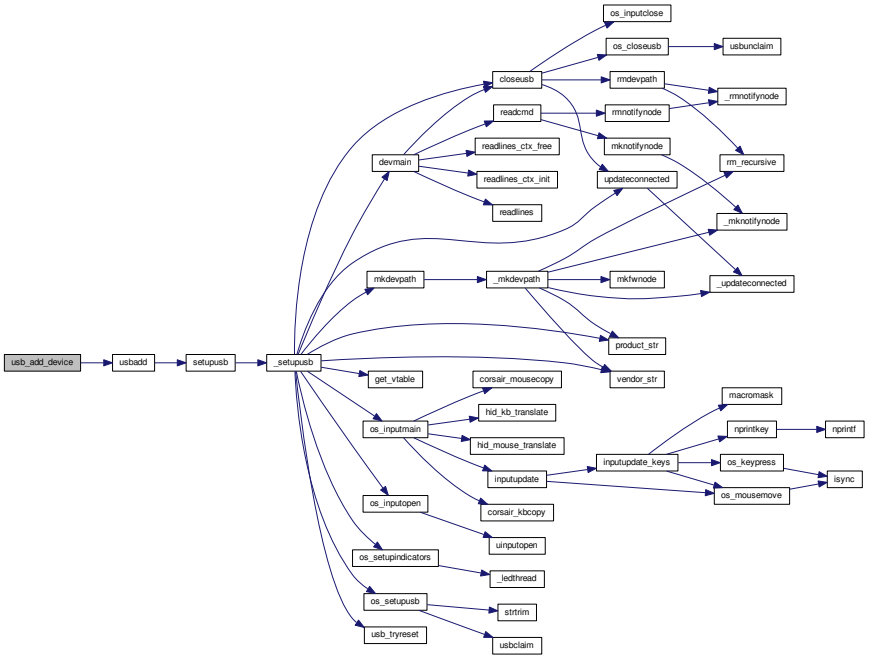
References N_MODELS, usbadd(), V_CORSAIR, and V_CORSAIR_STR.

Referenced by udev_enum(), and usbmain().

```
389                                             {
390     const char* vendor = udev_device_get_sysattr_value(dev, "idVendor");
391     if(vendor && !strcmp(vendor, V_CORSAIR_STR)){
```

```
392         const char* product = udev_device_get_sysattr_value(dev, "idProduct");
393         if(product){
394             for(_model* model = models; model < models +
    N_MODELS; model++){
395                 if(!strcmp(product, model->name)){
396                     return usbadd(dev, V_CORSAIR, model->number);
397                 }
398             }
399         }
400     }
401     return 1;
402 }
```

Here is the call graph for this function:



Here is the caller graph for this function:

8.43.3.12 static void usb_rm_device (struct udev_device * *dev*) [static]

Definition at line 405 of file usb_linux.c.

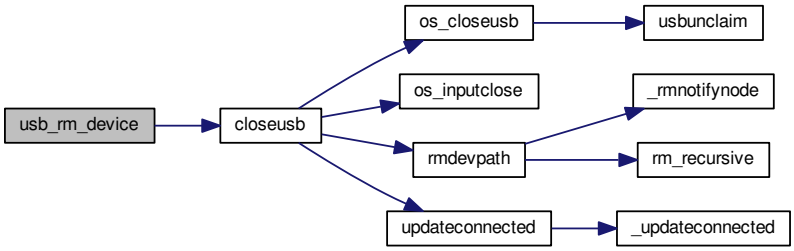
References closeusb(), DEV_MAX, devmutex, kbsyspath, and keyboard.

Referenced by usbmain().

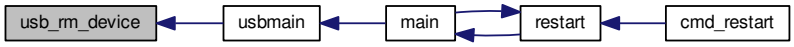
```
405                                             {
406     // Device removed. Look for it in our list of keyboards
407     const char* syspath = udev_device_get_syspath(dev);
408     if(!syspath || syspath[0] == 0)
409         return;
```

```
410     for(int i = 1; i < DEV_MAX; i++){
411         pthread_mutex_lock(devmutex + i);
412         if(!strcmp(syspath, kbsyspath[i]))
413             closeusb(keyboard + i);
414         pthread_mutex_unlock(devmutex + i);
415     }
416 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



8.43.3.13 int usbadd (struct udev_device * *dev*, short *vendor*, short *product*)

Definition at line 308 of file usb_linux.c.

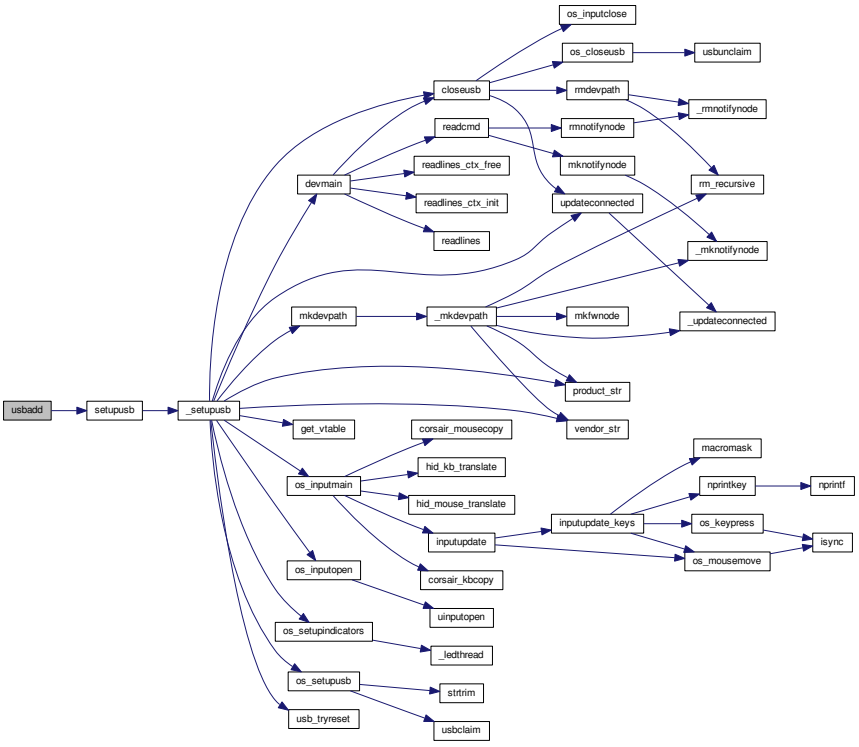
References ckb_err, DEV_MAX, dmutex, usbdevice::handle, IS_CONNECTED, kbsyspath, keyboard, usbdevice::product, setupusb(), usbdevice::udev, and usbdevice::vendor.

Referenced by usb_add_device().

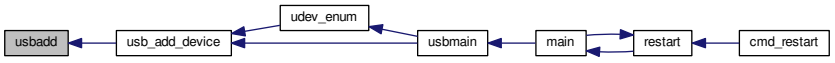
```
308                                                         {
309     const char* path = udev_device_get_devnode(dev);
310     const char* syspath = udev_device_get_syspath(dev);
311     if(!path || !syspath || path[0] == 0 || syspath[0] == 0){
312         ckb_err("Failed to get device path\n");
313         return -1;
314     }
315     // Find a free USB slot
316     for(int index = 1; index < DEV_MAX; index++){
317         usbdevice* kb = keyboard + index;
318         if(pthread_mutex_trylock(dmutex(kb))){
319             // If the mutex is locked then the device is obviously in use, so keep going
320             if(!strcmp(syspath, kbsyspath[index])){
321                 // Make sure this existing keyboard doesn't have the same syspath (this shouldn't happen)
322                 return 0;
323             }
324             continue;
325         }
326         if(!IS_CONNECTED(kb)){
327             // Open the sysfs device
328             kb->handle = open(path, O_RDWR) + 1;
329             if(kb->handle <= 0){
330                 ckb_err("Failed to open USB device: %s\n", strerror(errno));
331                 kb->handle = 0;
```

```
332                 pthread_mutex_unlock(dmutex(kb));
333                 return -1;
334             } else {
335                 // Set up device
336                 kb->udev = dev;
337                 kb->vendor = vendor;
338                 kb->product = product;
339                 strncpy(kbsyspath[index], syspath, FILENAME_MAX);
340                 // Mutex remains locked
341                 setupusb(kb);
342                 return 0;
343             }
344         }
345         pthread_mutex_unlock(dmutex(kb));
346     }
347     ckb_err("No free devices\n");
348     return -1;
349 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



8.43.3.14 int usbclaim (usbdevice * *kb*)

Definition at line 226 of file usb_linux.c.

References usbdevice::epcount, and usbdevice::handle.

Referenced by os_resetusb(), and os_setupusb().

```
226                             {
227     int count = kb->epcount;
228     for(int i = 0; i < count; i++){
229         struct usbdevfs_ioctl ctl = { i, USBDEVFS_DISCONNECT, 0 };
230         ioctl(kb->handle - 1, USBDEVFS_IOCTL, &ctl);
231         if(ioctl(kb->handle - 1, USBDEVFS_CLAIMINTERFACE, &i))
232             return -1;
233     }
234     return 0;
235 }
```

Here is the caller graph for this function:

8.43.3.15 void usbkill ( )

Definition at line 492 of file usb_linux.c.

Referenced by quitWithLock().

```
492             {
493     udev_unref(udev);
494     udev = 0;
495 }
```

Here is the caller graph for this function:

8.43.3.16 int usbmain ( )

Definition at line 441 of file usb_linux.c.

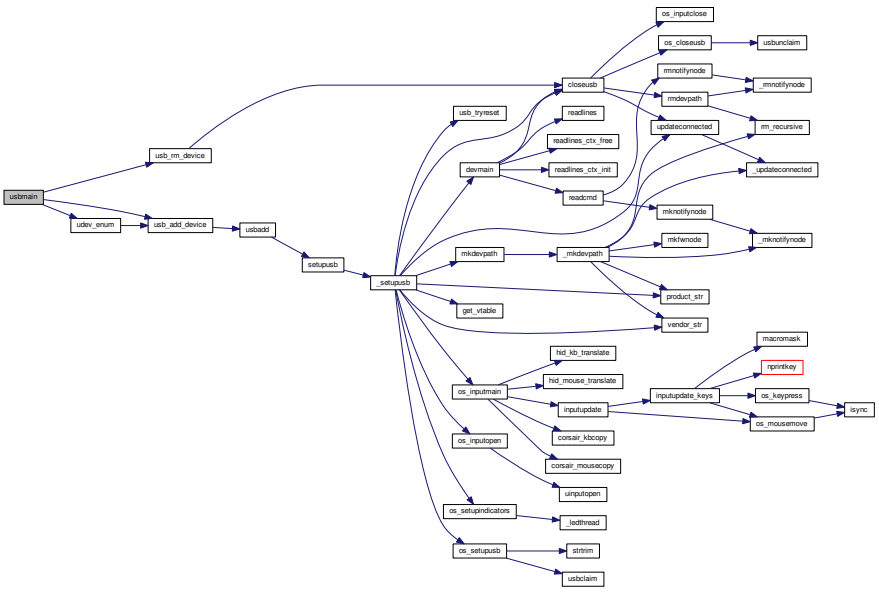
References ckb_fatal, ckb_warn, udev_enum(), usb_add_device(), and usb_rm_device().

Referenced by main().

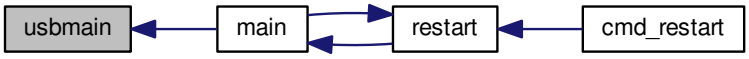
```
441             {
442     // Load the uinput module (if it's not loaded already)
443     if(system("modprobe uinput") != 0)
444         ckb_warn("Failed to load uinput module\n");
445
446     // Create the udev object
447     if(!(udev = udev_new())){
448         ckb_fatal("Failed to initialize udev\n");
449         return -1;
450     }
451
452     // Enumerate all currently connected devices
453     udev_enum();
454
455     // Done scanning. Enter a loop to poll for device updates
```

```
456     struct udev_monitor* monitor = udev_monitor_new_from_netlink(udev, "udev");
457     udev_monitor_filter_add_match_subsystem_devtype(monitor, "usb", 0);
458     udev_monitor_enable_receiving(monitor);
459     // Get an fd for the monitor
460     int fd = udev_monitor_get_fd(monitor);
461     fd_set fds;
462     while(udev){
463         FD_ZERO(&fds);
464         FD_SET(fd, &fds);
465         // Block until an event is read
466         if(select(fd + 1, &fds, 0, 0, 0) > 0 && FD_ISSET(fd, &fds)){
467             struct udev_device* dev = udev_monitor_receive_device(monitor);
468             if(!dev)
469                 continue;
470             const char* action = udev_device_get_action(dev);
471             if(!action){
472                 udev_device_unref(dev);
473                 continue;
474             }
475             // Add/remove device
476             if(!strcmp(action, "add")){
477                 int res = usb_add_device(dev);
478                 if(res == 0)
479                     continue;
480                 // If the device matched but the handle wasn't opened correctly, re-enumerate (this
    sometimes solves the problem)
481                 if(res == -1)
482                     udev_enum();
483             } else if(!strcmp(action, "remove"))
484                 usb_rm_device(dev);
485             udev_device_unref(dev);
486         }
487     }
488     udev_monitor_unref(monitor);
489     return 0;
490 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



8.43.3.17 int usbunclaim (usbdevice * kb, int resetting)

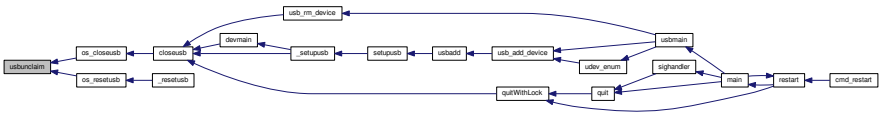
Definition at line 193 of file usb_linux.c.

References usbdevice::epcount, FEAT_RGB, usbdevice::handle, and HAS_FEATURES.

Referenced by os_closeusb(), and os_resetusb().

```
193                                             {
194     int handle = kb->handle - 1;
195     int count = kb->epcount;
196     for(int i = 0; i < count; i++)
197         ioctl(handle, USBDEVFS_RELEASEINTERFACE, &i);
198     // For RGB keyboards, the kernel driver should only be reconnected to interfaces 0 and 1 (HID), and
      only if we're not about to do a USB reset.
199     // Reconnecting any of the others causes trouble.
200     if(!resetting){
201         struct usbdevfs_ioctl ctl = { 0, USBDEVFS_CONNECT, 0 };
202         ioctl(handle, USBDEVFS_IOCTL, &ctl);
203         ctl.ifno = 1;
204         ioctl(handle, USBDEVFS_IOCTL, &ctl);
205         // Also reconnect iface #2 (HID) for non-RGB keyboards
206         if(!HAS_FEATURES(kb, FEAT_RGB)){
207             ctl.ifno = 2;
208             ioctl(handle, USBDEVFS_IOCTL, &ctl);
209         }
210     }
211     return 0;
212 }
```

Here is the caller graph for this function:



8.43.4 Variable Documentation

8.43.4.1 char kbsyspath[9][FILENAME_MAX] [static]

Definition at line 9 of file usb_linux.c.

Referenced by os_closeusb(), usb_rm_device(), and usbadd().

8.43.4.2 _model models[] [static]

Initial value:

```
= {

    {  "1b17" ,  0x1b17  },
    {  "1b07" ,  0x1b07  },
    {  "1b37" ,  0x1b37  },
    {  "1b39" ,  0x1b39  },
    {  "1b13" ,  0x1b13  },
    {  "1b09" ,  0x1b09  },
    {  "1b33" ,  0x1b33  },
    {  "1b36" ,  0x1b36  },
    {  "1b38" ,  0x1b38  },
    {  "1b3a" ,  0x1b3a  },
    {  "1b11" ,  0x1b11  },
    {  "1b08" ,  0x1b08  },
    {  "1b2d" ,  0x1b2d  },
    {  "1b20" ,  0x1b20  },
    {  "1b15" ,  0x1b15  },

    {  "1b12" ,  0x1b12  },
    {  "1b2e" ,  0x1b2e  },
    {  "1b14" ,  0x1b14   },
    {  "1b19" ,  0x1b19   },
    {  "1b2f" ,  0x1b2f   },
    {  "1b1e" ,  0x1b1e  },
    {  "1b3e" ,  0x1b3e  },
    {  "1b32" ,  0x1b32   }
}
```

Definition at line 359 of file usb_linux.c.

8.43.4.3 struct udev* udev [static]

Definition at line 351 of file usb_linux.c.

8.43.4.4 pthread_t udevthread

Definition at line 352 of file usb_linux.c.

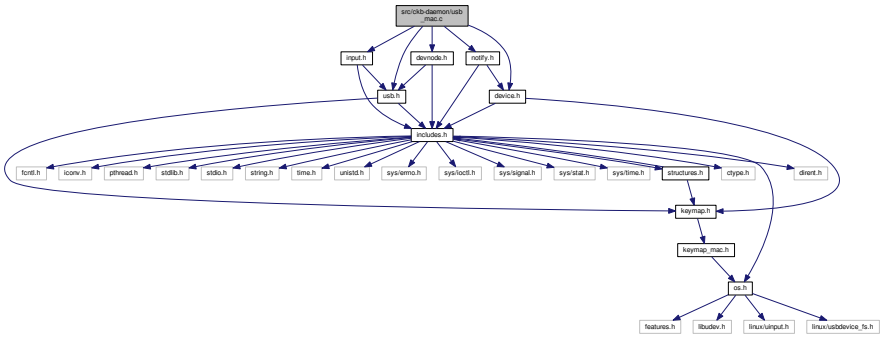
8.43.4.5 pthread_t usbthread

Definition at line 352 of file usb_linux.c.

8.44 src/ckb-daemon/usb_mac.c File Reference

```
#include "device.h"
#include "devnode.h"
#include "input.h"
#include "notify.h"
#include "usb.h"
```

Include dependency graph for usb_mac.c:



Index