ckb-next

0.2.7

Generated by Doxygen 1.8.6

Thu Apr 27 2017 21:14:34

Contents

Chapter 1

Todo List

Global _usbsend (usbdevice ∗kb, const uchar ∗messages, int count, const char ∗file, int line)

A lot of different conditions are combined in this code. Don't think, it is good in every combination...

Check whether this is the same in the macOS variant. It is not dramatic, but if errors occur, it can certainly irritate the devices completely if they receive incomplete data streams. Do we have errors with the messages "Wrote YY bytes (expected 64)" in the system logs? If not, we do not need to look any further.

Global closeusb (usbdevice ∗kb)

What is not yet comprehensible is the call to updateconnected() BEFORE os_closeusb(). Should that be in the other sequence? Or is updateconnected() not displaying the connected usb devices, but the representation which uinput devices are loaded? Questions about questions ...

Global devmain (usbdevice ∗kb)

Hope to find the need for dmutex usage later.

Should this function be declared as pthread_t∗ function, because of the defintion of pthread-create? But void∗ works also...

readcmd() gets a **line**, not **lines**. Have a look on that later.

Is the condition IS_CONNECTED valid? What functions change the condititon for the macro?

Global get_vtable (short vendor, short product)

Is the last point really a good decision and always correct?

Global os_inputmain (void ∗context)

This function is a collection of many tasks. It should be divided into several sub-functions for the sake of greater convenience:

Global os_resetusb (usbdevice ∗kb, const char ∗file, int line)

it seems that no one wants to try the reset again. But I'v seen it somewhere...

Global os_setupusb (usbdevice ∗kb)

in these modules a pullrequest is outstanding

Global os_usbsend (usbdevice ∗kb, const uchar ∗out_msg, int is_recv, const char ∗file, int line)

Since the handling of endpoints has already led to problems elsewhere, this implementation is extremely hardware-dependent and critical!

Eg. the new keyboard K95PLATINUMRGB has a version number significantly less than 2.0 - will it run with this implementation?

Global product_str (short product)

There are macros defined in usb.h to detect all the combinations below. the only difference is the parameter: The macros need the *kb∗*, product_str() needs the *product ID*

Global revertusb (usbdevice ∗kb)

Why is this useful? Are there problems seen with deactivating a device with older fw-version??? Why isn't this an error indicating reason and we return success (0)?

The return value of nk95cmd() is ignored (but sending the ioctl may produce an error and _nk95_cmd will indicate this), instead revertusb() returns success in any case.

Global udevthread

These two thread vasriables seem to be unused: usbtread, udevthread

Global udevthread

These two thread vasriables seem to be unused: usbtread, udevthread

Global usb_add_device (struct udev_device ∗dev)

So why the hell not a transformation between the string and the short presentation? Lets check if the string representation is used elsewhere.

Global usb_tryreset (usbdevice ∗kb)

Why does usb_tryreset() hide the information returned from resetusb()? Isn't it needed by the callers?

Global usbmain ()

Why isn't missing of uinput a fatal error?

lae. here the work has to go on...

Global usbmutex

We should have a look why this mutex is never used.

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 3

File Index

3.1 File List

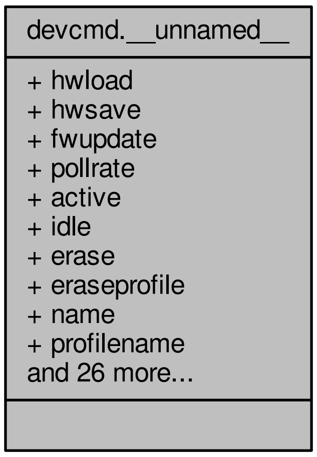
Here is a list of all files with brief descriptions:

Chapter 4

Data Structure Documentation

4.1 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)

4.1.1 Detailed Description

Definition at line 78 of file command.h.

4.1.2 Field Documentation

4.1.2.1

4.1.2.2

4.1.2.3

4.1.2.4

4.1.2.5

4.1.2.6

4.1.2.7

4.1.2.8

4.1.2.9

4.1.2.10

4.1.2.11

4.1.2.12

4.1.2.13

4.1.2.14

4.1.2.15

4.1.2.16

4.1.2.17

4.1.2.18

4.1.2.19

4.1.2.20

4.1.2.21

4.1.2.22

4.1.2.23

4.1.2.24

4.1.2.25

4.1.2.26

4.1.2.27

4.1.2.28

4.1.2.29

4.1.2.30

4.1.2.31

4.1.2.32

4.1.2.33

4.1.2.34

4.1.2.35

4.1.2.36

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

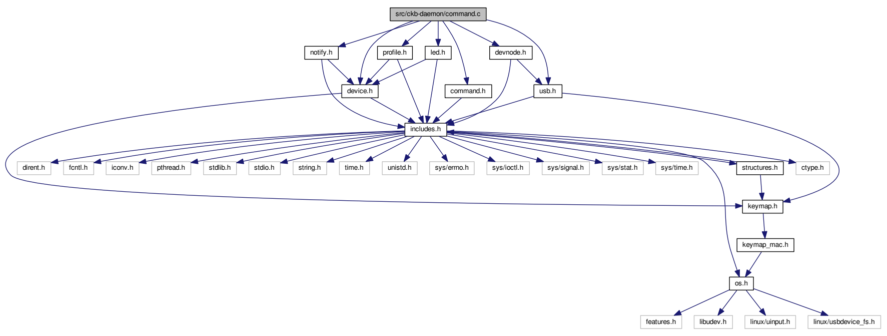
Chapter 5

File Documentation

5.1 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]

5.1.1 Macro Definition Documentation

5.1.1.1 #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().

5.1.2 Function Documentation

5.1.2.1 int readcmd (usbdevice ∗ *kb,* const char ∗ *line*)

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

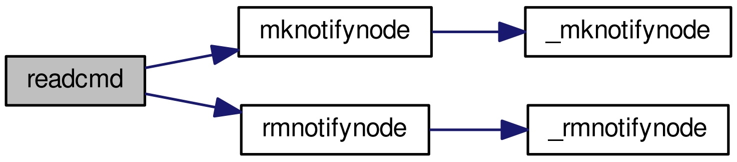
Definition at line 67 of file command.c.

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.

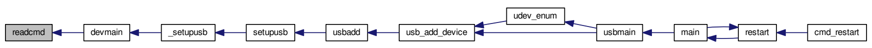
Referenced by devmain().

```
                                               {
    char* word = malloc(strlen(line) + 1);
    int wordlen;
    const char* newline = 0;
    const devcmd* vt = kb->vtable;
    usbprofile* profile = kb->profile;
    usbmode* mode = 0;
    int notifynumber = 0;
    // Read words from the input
    cmd command = NONE;
    while(sscanf(line, "%s%n", word, &wordlen) == 1){
        line += wordlen;
        // If we passed a newline, reset the context
        if(line > newline){
            mode = profile->currentmode;
            command = NONE;
            notifynumber = 0;
            newline = strchr(line, '\n');
            if(!newline)
                newline = line + strlen(line);
        }
        // Check for a command word
        for(int i = 0; i < CMD_COUNT - 1; i++){
            if(!strcmp(word, cmd_strings[i])){
                command = i + CMD_FIRST;
#ifndef OS_MAC
                // Layout and mouse acceleration aren't used on Linux; ignore
                if(command == LAYOUT || command == ACCEL || command ==
   SCROLLSPEED)
                    command = NONE;
#endif
                // Most commands require parameters, but a few are actions in and of themselves
                if(command != SWITCH
                        && command != HWLOAD && command != HWSAVE
                        && command != ACTIVE && command != IDLE
                        && command != ERASE && command != ERASEPROFILE
                        && command != RESTART)
                    goto next_loop;
                break;
            }
        }

        // Set current notification node when given @number
        int newnotify;
        if(sscanf(word, "@%u", &newnotify) == 1 && newnotify < OUTFIFO_MAX){
            notifynumber = newnotify;
            continue;
        }

        // Reject unrecognized commands. Reject bind or notify related commands if the keyboard doesn't have the feature enabled.
        if(command == NONE
                || ((!HAS_FEATURES(kb, FEAT_BIND) && (command == BIND || command == UNBIND || command == REBIND || command == MACRO || command == DELAY))
                           || (!HAS_FEATURES(kb, FEAT_NOTIFY) && command == NOTIFY))){
            next_loop:
            continue;
        }
        // Reject anything not related to fwupdate if device has a bricked FW
        if(NEEDS_FW_UPDATE(kb) && command != FWUPDATE && command != NOTIFYON && command != NOTIFYOFF)
            continue;

        // Specially handled commands - these are available even when keyboard is IDLE
        switch(command){
        case NOTIFYON: {
            // Notification node on
            int notify;
            if(sscanf(word, "%u", &notify) == 1)
                mknotifynode(kb, notify);
            continue;
        } case NOTIFYOFF: {
            // Notification node off
            int notify;
            if(sscanf(word, "%u", &notify) == 1 && notify != 0) // notify0 can't be removed
                rmnotifynode(kb, notify);
            continue;
        } case GET:
            // Output data to notification node
            vt->get(kb, mode, notifynumber, 0, word);
            continue;
        case LAYOUT:
            // OSX: switch ANSI/ISO keyboard layout
            if(!strcmp(word, "ansi"))
                kb->features = (kb->features & ~FEAT_LMASK) | FEAT_ANSI;
            else if(!strcmp(word, "iso"))
                kb->features = (kb->features & ~FEAT_LMASK) | FEAT_ISO;
            continue;
#ifdef OS_MAC
        case ACCEL:
            // OSX mouse acceleration on/off
            if(!strcmp(word, "on"))
                kb->features |= FEAT_MOUSEACCEL;
            else if(!strcmp(word, "off"))
                kb->features &= ~FEAT_MOUSEACCEL;
            continue;
        case SCROLLSPEED:{
            int newscroll;
            if(sscanf(word, "%d", &newscroll) != 1)
                break;
            if(newscroll < SCROLL_MIN)
                newscroll = SCROLL_ACCELERATED;
            if(newscroll > SCROLL_MAX)
                newscroll = SCROLL_MAX;
            kb->scroll_rate = newscroll;
            continue;
        }
#endif
        case MODE: {
            // Select a mode number (1 - 6)
            int newmode;
            if(sscanf(word, "%u", &newmode) == 1 && newmode > 0 && newmode <= MODE_COUNT)
                mode = profile->mode + newmode - 1;
            continue;
        }
        case FPS: {
            // USB command delay (2 - 10ms)
            uint framerate;
            if(sscanf(word, "%u", &framerate) == 1 && framerate > 0){
                // Not all devices require the same number of messages per frame; select delay
appropriately
                uint per_frame = IS_MOUSE_DEV(kb) ? 2 : IS_FULLRANGE(kb) ? 14 : 5;
                uint delay = 1000 / framerate / per_frame;
                if(delay < 2)
                    delay = 2;
                else if(delay > 10)
                    delay = 10;
                kb->usbdelay = delay;
            }
            continue;
        }
        case DITHER: {
            // 0: No dither, 1: Ordered dither.
            uint dither;
            if(sscanf(word, "%u", &dither) == 1 && dither <= 1){
                kb->dither = dither;
                profile->currentmode->light.forceupdate = 1;
                mode->light.forceupdate = 1;
            }
            continue;
        }
        case DELAY:
            kb->delay = (!strcmp (word, "on")); // independendant from parameter to handle false
commands like "delay off"
            continue;
        case RESTART: {
            char mybuffer[] = "no reason specified";
            if (sscanf(line, " %[^\n]", word) == -1) {
                word = mybuffer;
            }
            vt->do_cmd[command](kb, mode, notifynumber, 0, word);
            continue;
        }

        default:;
        }

        // If a keyboard is inactive, it must be activated before receiving any other commands
        if(!kb->active){
            if(command == ACTIVE)
                TRY_WITH_RESET(vt->active(kb, mode, notifynumber, 0, 0));
            continue;
        }
        // Specially handled commands only available when keyboard is ACTIVE
        switch(command){
        case IDLE:
            TRY_WITH_RESET(vt->idle(kb, mode, notifynumber, 0, 0));
            continue;
        case SWITCH:
            if(profile->currentmode != mode){
                profile->currentmode = mode;
                // Set mode light for non-RGB K95
                int index = INDEX_OF(mode, profile->mode);
                vt->setmodeindex(kb, index);
            }
            continue;
        case HWLOAD: case HWSAVE:{
            char delay = kb->usbdelay;
            // Ensure delay of at least 10ms as the device can get overwhelmed otherwise
            if(delay < 10)
                kb->usbdelay = 10;
            // Try to load/save the hardware profile. Reset on failure, disconnect if reset fails.
            TRY_WITH_RESET(vt->do_io[command](kb, mode, notifynumber, 1, 0));
            // Re-send the current RGB state as it sometimes gets scrambled
            TRY_WITH_RESET(vt->updatergb(kb, 1));
            kb->usbdelay = delay;
            continue;
        }
        case FWUPDATE:
            // FW update parses a whole word. Unlike hwload/hwsave, there's no try again on failure.
            if(vt->fwupdate(kb, mode, notifynumber, 0, word)){
                free(word);
                return 1;
            }
            continue;
        case POLLRATE: {
            uint rate;
            if(sscanf(word, "%u", &rate) == 1 && (rate == 1 || rate == 2 || rate == 4 || rate == 8))
                TRY_WITH_RESET(vt->pollrate(kb, mode, notifynumber, rate, 0));
            continue;
        }
        case ERASEPROFILE:
            // Erase the current profile
            vt->eraseprofile(kb, mode, notifynumber, 0, 0);
            // Update profile/mode pointers
            profile = kb->profile;
            mode = profile->currentmode;
            continue;
        case ERASE: case NAME: case IOFF: case ION: case IAUTO: case
    INOTIFY: case PROFILENAME: case ID: case PROFILEID: case
    DPISEL: case LIFT: case SNAP:
            // All of the above just parse the whole word
            vt->do_cmd[command](kb, mode, notifynumber, 0, word);
            continue;
        case RGB: {
            // RGB command has a special response for a single hex constant
            int r, g, b;
            if(sscanf(word, "%02x%02x%02x", &r, &g, &b) == 3){
                // Set all keys
                for(int i = 0; i < N_KEYS_EXTENDED; i++)
                    vt->rgb(kb, mode, notifynumber, i, word);
                continue;
            }
            break;
        }
        case MACRO:
            if(!strcmp(word, "clear")){
                // Macro has a special clear command
                vt->macro(kb, mode, notifynumber, 0, 0);
                continue;
            }
            break;
        default:;
        }
        // For anything else, split the parameter at the colon
        int left = -1;
        sscanf(word, "%*[^:]%n", &left);
        if(left <= 0)
            continue;
        const char* right = word + left;
        if(right[0] == ':')
            right++;
        // Macros and DPI have a separate left-side handler
        if(command == MACRO || command == DPI){
            word[left] = 0;
            vt->do_macro[command](kb, mode, notifynumber, word, right);
            continue;
        }
        // Scan the left side for key names and run the requested command
        int position = 0, field = 0;
        char keyname[11];
        while(position < left && sscanf(word + position, "%10[^:,]%n", keyname, &field) == 1){
            int keycode;
            if(!strcmp(keyname, "all")){
                // Set all keys
                for(int i = 0; i < N_KEYS_EXTENDED; i++)
                    vt->do_cmd[command](kb, mode, notifynumber, i, right);
            } else if((sscanf(keyname, "#%d", &keycode) && keycode >= 0 && keycode <
    N_KEYS_EXTENDED)
                      || (sscanf(keyname, "#x%x", &keycode) && keycode >= 0 && keycode <
    N_KEYS_EXTENDED)){
                // Set a key numerically
                vt->do_cmd[command](kb, mode, notifynumber, keycode, right);
            } else {
                // Find this key in the keymap
                for(unsigned i = 0; i < N_KEYS_EXTENDED; i++){
                    if(keymap[i].name && !strcmp(keyname, keymap[i].name)){
                        vt->do_cmd[command](kb, mode, notifynumber, i, right);
                        break;
                    }
                }
            }
            if(word[position += field] == ',')
                position++;
        }
    }

    // Finish up
    if(!NEEDS_FW_UPDATE(kb)){
        TRY_WITH_RESET(vt->updatergb(kb, 0));
        TRY_WITH_RESET(vt->updatedpi(kb, 0));
    }
    free(word);
    return 0;
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.1.3 Variable Documentation

5.1.3.1 const char∗ const cmd_strings[(CMD_LAST-CMD_FIRST+2)-1] `[static]`

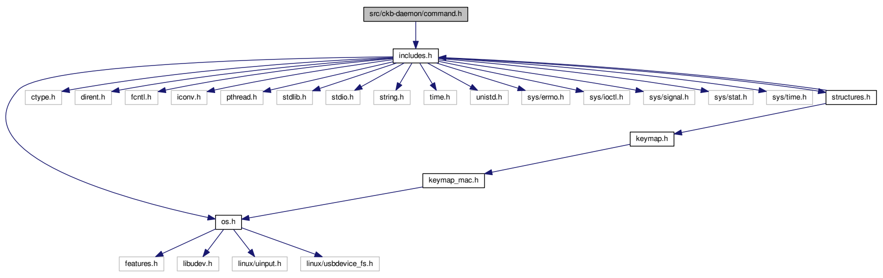
Definition at line 9 of file command.c.

Referenced by readcmd().

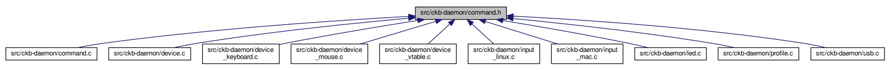
5.2 src/ckb-daemon/command.h File Reference

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

Include dependency graph for command.h:



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


src/ckb-daemon/command.h
src/ckb-daemon/command.c
src/ckb-daemon/device.c
src/ckb-daemon/device _keyboard.c
src/ckb-daemon/device _mouse.c
src/ckb-daemon/device _vtable.c
src/ckb-daemon/input _linux.c
src/ckb-daemon/input _mac.c
src/ckb-daemon/led.c
src/ckb-daemon/profile.c
src/ckb-daemon/usb.c

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

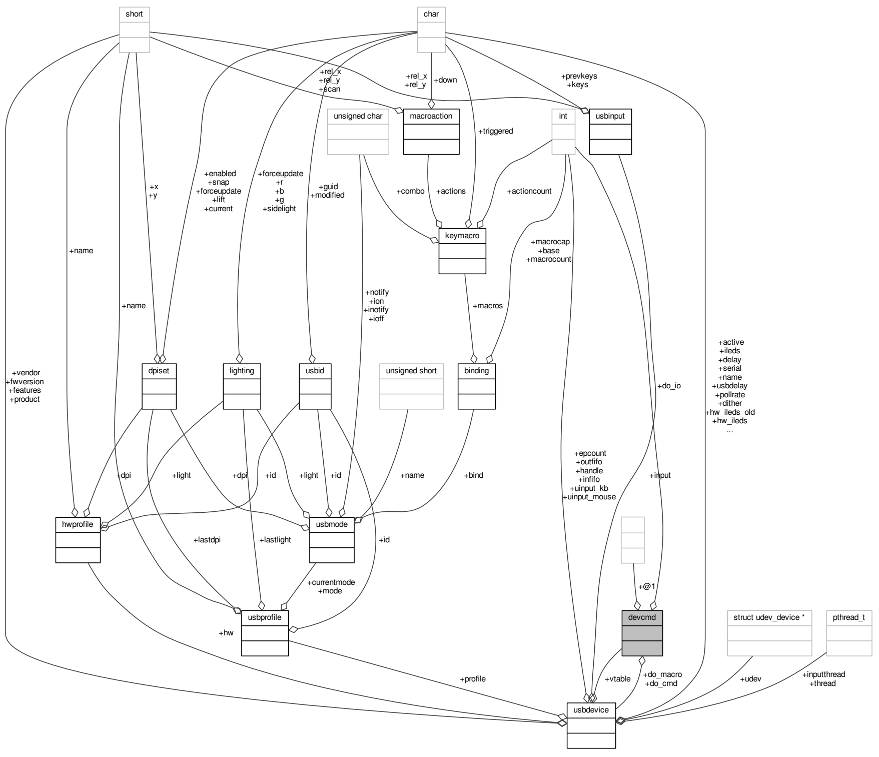
- int readcmd (usbdevice ∗kb, const char ∗line)

5.2.1 Data Structure Documentation

5.2.1.1 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)]	

5.2.2 Macro Definition Documentation

5.2.2.1 #define CMD_COUNT (CMD_LAST - CMD_FIRST + 2)

Definition at line 65 of file command.h.

Referenced by readcmd().

5.2.2.2 #define CMD_DEV_COUNT (CMD_LAST - CMD_VT_FIRST + 1)

Definition at line 66 of file command.h.

5.2.3 Typedef Documentation

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

Definition at line 70 of file command.h.

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

Definition at line 71 of file command.h.

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

Definition at line 72 of file command.h.

5.2.3.4 typedef union devcmd devcmd

5.2.4 Enumeration Type Documentation

5.2.4.1 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.

```
             {
    // Special - handled by readcmd, no device functions
    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,

     // Hardware data
     HWLOAD      = 0,    CMD_VT_FIRST = 0,
     HWSAVE,
     FWUPDATE,
     POLLRATE,

     // Software control on/off
     ACTIVE,
     IDLE,

     // Profile/mode metadata
     ERASE,
     ERASEPROFILE,
     NAME,
     PROFILENAME,
     ID,
     PROFILEID,

     // LED control
     RGB,
     IOFF,
     ION,
     IAUTO,

     // Key binding control
     BIND,
     UNBIND,
     REBIND,
     MACRO,

     // DPI control
     DPI,
     DPISEL,
     LIFT,
     SNAP,

     // Notifications and output
     NOTIFY,
     INOTIFY,
60     GET,
61     RESTART,
62
63     CMD_LAST = RESTART
64 } cmd;
```

5.2.5 Function Documentation

5.2.5.1 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.

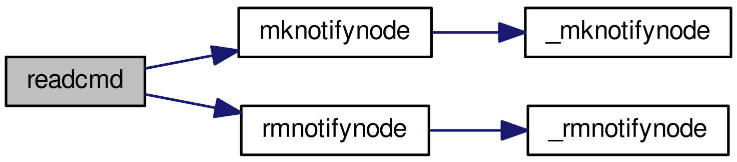
References ACCEL, ACTIVE, usbdevice::active, BIND, CMD_COUNT, CMD_FIRST, cmd_strings, usbprofile::currentmode, DELAY, usbdevice::delay, DITHER, usbdevice::dither, devcmd::do_cmd, devcmd::do_io, devcmd::do_macro, DPI, DPISEL, ERASE, ERASEPROFILE, FEAT_ANSI, FEAT_BIND, FEAT_ISO, FEAT_LMASK, FEAT_MOUSEACCEL, 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().

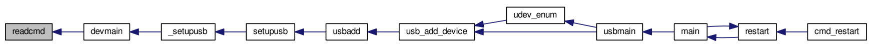
```
67                                             {
68     char* word = malloc(strlen(line) + 1);
69     int wordlen;
70     const char* newline = 0;
71     const devcmd* vt = kb->vtable;
72     usbprofile* profile = kb->profile;
73     usbmode* mode = 0;
74     int notifynumber = 0;
75     // Read 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
```

```
        // Reject unrecognized commands. Reject bind or notify related commands if the keyboard doesn't have the feature enabled.
        if(command == NONE
                || ((!HAS_FEATURES(kb, FEAT_BIND) && (command == BIND || command == UNBIND || command == REBIND || command == MACRO || command == DELAY))
                           || (!HAS_FEATURES(kb, FEAT_NOTIFY) && command == NOTIFY))){
            next_loop:
            continue;
        }
        // Reject anything not related to fwupdate if device has a bricked FW
        if(NEEDS_FW_UPDATE(kb) && command != FWUPDATE && command != NOTIFYON && command != NOTIFYOFF)
            continue;

        // Specially handled commands - these are available even when keyboard is IDLE
        switch(command){
        case NOTIFYON: {
            // Notification node on
            int notify;
            if(sscanf(word, "%u", &notify) == 1)
                mknotifynode(kb, notify);
            continue;
        } case NOTIFYOFF: {
            // Notification node off
            int notify;
            if(sscanf(word, "%u", &notify) == 1 && notify != 0) // notify0 can't be removed
                rmnotifynode(kb, notify);
            continue;
        } case GET:
            // Output data to notification node
            vt->get(kb, mode, notifynumber, 0, word);
            continue;
        case LAYOUT:
            // OSX: switch ANSI/ISO keyboard layout
            if(!strcmp(word, "ansi"))
                kb->features = (kb->features & ~FEAT_LMASK) | FEAT_ANSI;
            else if(!strcmp(word, "iso"))
                kb->features = (kb->features & ~FEAT_LMASK) | FEAT_ISO;
            continue;
#ifdef OS_MAC
        case ACCEL:
            // OSX mouse acceleration on/off
            if(!strcmp(word, "on"))
                kb->features |= FEAT_MOUSEACCEL;
            else if(!strcmp(word, "off"))
                kb->features &= ~FEAT_MOUSEACCEL;
            continue;
        case SCROLLSPEED:{
            int newscroll;
            if(sscanf(word, "%d", &newscroll) != 1)
                break;
            if(newscroll < SCROLL_MIN)
                newscroll = SCROLL_ACCELERATED;
            if(newscroll > SCROLL_MAX)
                newscroll = SCROLL_MAX;
            kb->scroll_rate = newscroll;
            continue;
        }
#endif
        case MODE: {
            // Select a mode number (1 - 6)
            int newmode;
            if(sscanf(word, "%u", &newmode) == 1 && newmode > 0 && newmode <= MODE_COUNT)
                mode = profile->mode + newmode - 1;
            continue;
        }
        case FPS: {
            // USB command delay (2 - 10ms)
            uint framerate;
            if(sscanf(word, "%u", &framerate) == 1 && framerate > 0){
                // Not all devices require the same number of messages per frame; select delay appropriately
                uint per_frame = IS_MOUSE_DEV(kb) ? 2 : IS_FULLRANGE(kb) ? 14 : 5;
                uint delay = 1000 / framerate / per_frame;
                if(delay < 2)
                    delay = 2;
                else if(delay > 10)
                    delay = 10;
                kb->usbdelay = delay;
            }
            continue;
        }
        case DITHER: {
            // 0: No dither, 1: Ordered dither.
            uint dither;
            if(sscanf(word, "%u", &dither) == 1 && dither <= 1){
                kb->dither = dither;
                profile->currentmode->light.forceupdate = 1;
                mode->light.forceupdate = 1;
            }
            continue;
        }
        case DELAY:
            kb->delay = (!strcmp (word, "on")); // independendant from parameter to handle false commands like "delay off"
            continue;
        case RESTART: {
            char mybuffer[] = "no reason specified";
            if (sscanf(line, " %[^\n]", word) == -1) {
                word = mybuffer;
            }
            vt->do_cmd[command](kb, mode, notifynumber, 0, word);
            continue;
        }

        default:;
        }

        // If a keyboard is inactive, it must be activated before receiving any other commands
        if(!kb->active){
            if(command == ACTIVE)
                TRY_WITH_RESET(vt->active(kb, mode, notifynumber, 0, 0));
            continue;
        }
        // Specially handled commands only available when keyboard is ACTIVE
        switch(command){
        case IDLE:
            TRY_WITH_RESET(vt->idle(kb, mode, notifynumber, 0, 0));
            continue;
        case SWITCH:
            if(profile->currentmode != mode){
                profile->currentmode = mode;
                // Set mode light for non-RGB K95
                int index = INDEX_OF(mode, profile->mode);
                vt->setmodeindex(kb, index);
            }
            continue;
        case HWLOAD: case HWSAVE:{
            char delay = kb->usbdelay;
            // Ensure delay of at least 10ms as the device can get overwhelmed otherwise
            if(delay < 10)
                kb->usbdelay = 10;
            // Try to load/save the hardware profile. Reset on failure, disconnect if reset fails.
            TRY_WITH_RESET(vt->do_io[command](kb, mode, notifynumber, 1, 0));
            // Re-send the current RGB state as it sometimes gets scrambled
            TRY_WITH_RESET(vt->updatergb(kb, 1));
            kb->usbdelay = delay;
            continue;
        }
        case FWUPDATE:
            // FW update parses a whole word. Unlike hwload/hwsave, there's no try again on failure.
            if(vt->fwupdate(kb, mode, notifynumber, 0, word)){
                free(word);
                return 1;
            }
            continue;
        case POLLRATE: {
            uint rate;
            if(sscanf(word, "%u", &rate) == 1 && (rate == 1 || rate == 2 || rate == 4 || rate == 8))
                TRY_WITH_RESET(vt->pollrate(kb, mode, notifynumber, rate, 0));
            continue;
        }
        case ERASEPROFILE:
            // Erase the current profile
            vt->eraseprofile(kb, mode, notifynumber, 0, 0);
            // Update profile/mode pointers
            profile = kb->profile;
            mode = profile->currentmode;
            continue;
        case ERASE: case NAME: case IOFF: case ION: case IAUTO: case INOTIFY: case PROFILENAME: case ID: case PROFILEID: case DPISEL: case LIFT: case SNAP:
            // All of the above just parse the whole word
            vt->do_cmd[command](kb, mode, notifynumber, 0, word);
            continue;
        case RGB: {
            // RGB command has a special response for a single hex constant
            int r, g, b;
            if(sscanf(word, "%02x%02x%02x", &r, &g, &b) == 3){
                // Set all keys
                for(int i = 0; i < N_KEYS_EXTENDED; i++)
                    vt->rgb(kb, mode, notifynumber, i, word);
                continue;
            }
            break;
        }
        case MACRO:
            if(!strcmp(word, "clear")){
                // Macro has a special clear command
                vt->macro(kb, mode, notifynumber, 0, 0);
                continue;
            }
            break;
        default:;
        }
        // For anything else, split the parameter at the colon
        int left = -1;
        sscanf(word, "%*[^:]%n", &left);
        if(left <= 0)
            continue;
        const char* right = word + left;
        if(right[0] == ':')
            right++;
        // Macros and DPI have a separate left-side handler
        if(command == MACRO || command == DPI){
            word[left] = 0;
            vt->do_macro[command](kb, mode, notifynumber, word, right);
            continue;
        }
        // Scan the left side for key names and run the requested command
        int position = 0, field = 0;
        char keyname[11];
        while(position < left && sscanf(word + position, "%10[^:,]%n", keyname, &field) == 1){
            int keycode;
            if(!strcmp(keyname, "all")){
                // Set all keys
                for(int i = 0; i < N_KEYS_EXTENDED; i++)
                    vt->do_cmd[command](kb, mode, notifynumber, i, right);
            } else if((sscanf(keyname, "#%d", &keycode) && keycode >= 0 && keycode <
  N_KEYS_EXTENDED)
                      || (sscanf(keyname, "#x%x", &keycode) && keycode >= 0 && keycode <
  N_KEYS_EXTENDED)){
                // Set a key numerically
                vt->do_cmd[command](kb, mode, notifynumber, keycode, right);
            } else {
                // Find this key in the keymap
                for(unsigned i = 0; i < N_KEYS_EXTENDED; i++){
                    if(keymap[i].name && !strcmp(keyname, keymap[i].name)){
                        vt->do_cmd[command](kb, mode, notifynumber, i, right);
                        break;
                    }
                }
            }
            if(word[position += field] == ',')
                position++;
        }
    }

    // Finish up
    if(!NEEDS_FW_UPDATE(kb)){
        TRY_WITH_RESET(vt->updatergb(kb, 0));
        TRY_WITH_RESET(vt->updatedpi(kb, 0));
    }
    free(word);
    return 0;
}
```

Here is the call graph for this function:



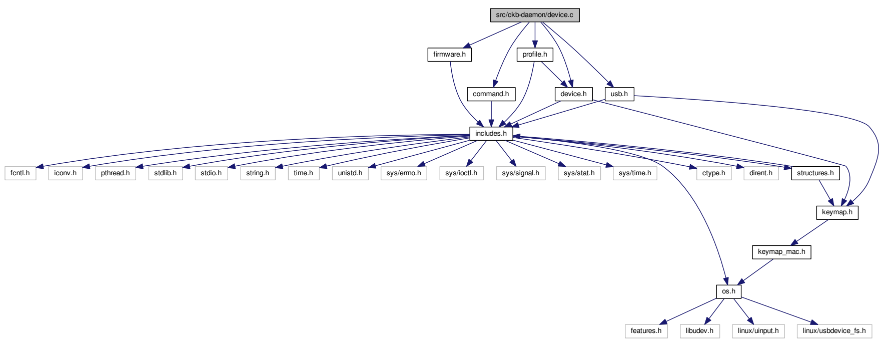
Here is the caller graph for this function:



5.3 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]

hwload_mode = 1 means read hardware once. should be enough

- pthread_mutex_t devlistmutex = PTHREAD_MUTEX_INITIALIZER

  remember all usb devices. Needed for closeusb().
- pthread_mutex_t devmutex [9] = { [0 ... 9 -1] = PTHREAD_MUTEX_INITIALIZER }
- pthread_mutex_t inputmutex [9] = { [0 ... 9 -1] = PTHREAD_MUTEX_INITIALIZER }

5.3.1 Function Documentation

5.3.1.1 int _start_dev (**usbdevice** ∗ *kb,* int *makeactive*)

_start_dev get fw-info and pollrate; if available, install new firmware; get all hardware profiles

Parameters

kb	the normal kb pointer to the usbdevice. Is also valid for mice.
makeactive	if set to 1, activate the device via setactive()

Returns

0 if success, other else

- This hacker code is tricky in mutliple aspects. What it means is:

  if hwload_mode == 0: just set pollrate to 0 and clear features in the bottom lines of the if-block.

  if hwload_mode == 1: if the device has FEAT_HWLOAD active, call getfwversion(). If it returns true, there was an error while detecting fw-version. Put error message, reset FEAT_HWLOAD and finalize as above.

  if hwload_mode == 2: if the device has FEAT_HWLOAD active, call getfwversion(). If it returns true, there was an error while detecting fw-version. Put error message and return directly from function with error.

  Why do not you just write it down?
- Now check if device needs a firmware update. If so, set it up and leave the function without error.
- Device needs a firmware update. Finish setting up but don't do anything.
- Load profile from device if the hw-pointer is not set yet and hw-loading is possible and allowed.

  return error if mode == 2 (load always) and loading got an error. Else reset HWLOAD feature, because hwload must be 1.

  That is real Horror code.

Definition at line 22 of file device.c.

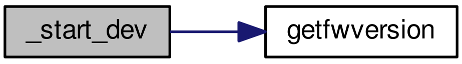
References usbdevice::active, ckb_info, ckb_warn, FEAT_ADJRATE, FEAT_FWUPDATE, FEAT_FWVERSION, FEAT_HWLOAD, FEAT_POLLRATE, FEAT_RGB, usbdevice::features, usbdevice::fwversion, getfwversion(), HAS_FEATURES, usbdevice::hw, hwload_mode, hwloadprofile, NEEDS_FW_UPDATE, usbdevice::pollrate, and setactive.

Referenced by start_dev().

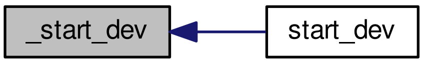
```
22                                                {
23     // Get the firmware version from the device
24     if(kb->pollrate == 0){
32         if(!hwload_mode || (HAS_FEATURES(kb, FEAT_HWLOAD) &&
   getfwversion(kb))){
33             if(hwload_mode == 2)
34                 // hwload=always. Report setup failure.
35                 return -1;
36             else if(hwload_mode){
37                 // hwload=once. Log failure, prevent trying again, and continue.
38                 ckb_warn("Unable to load firmware version/poll rate\n");
39                 kb->features &= ~FEAT_HWLOAD;
40             }
41             kb->pollrate = 0;
42             kb->features &= ~(FEAT_POLLRATE | FEAT_ADJRATE);
43             if(kb->fwversion == 0)
```

```
                kb->features &= ~(FEAT_FWVERSION |
    FEAT_FWUPDATE);
            }
        }
        if(NEEDS_FW_UPDATE(kb)){
            ckb_info("Device needs a firmware update. Please issue a fwupdate command.\n");
            kb->features = FEAT_RGB | FEAT_FWVERSION |
    FEAT_FWUPDATE;
            kb->active = 1;
            return 0;
        }
        if(!kb->hw && hwload_mode && HAS_FEATURES(kb,
    FEAT_HWLOAD)){
            if(hwloadprofile(kb, 1)){
                if(hwload_mode == 2)
                    return -1;
                ckb_warn("Unable to load hardware profile\n");
                kb->features &= ~FEAT_HWLOAD;
            }
        }
        // Active software mode if requested
        if(makeactive)
            return setactive(kb, 1);
        return 0;
    }
```

Here is the call graph for this function:



Here is the caller graph for this function:



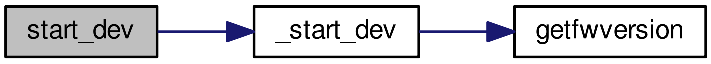
5.3.1.2 int start_dev (usbdevice ∗ *kb,* int *makeactive*)

Definition at line 77 of file device.c.

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

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

Here is the call graph for this function:



5.3.2 Variable Documentation

5.3.2.1 pthread_mutex_t devlistmutex = PTHREAD_MUTEX_INITIALIZER

Definition at line 11 of file device.c.

5.3.2.2 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().

5.3.2.3 int hwload_mode = 1

Definition at line 7 of file device.c.

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

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

Definition at line 13 of file device.c.

5.3.2.5 usbdevice keyboard[9]

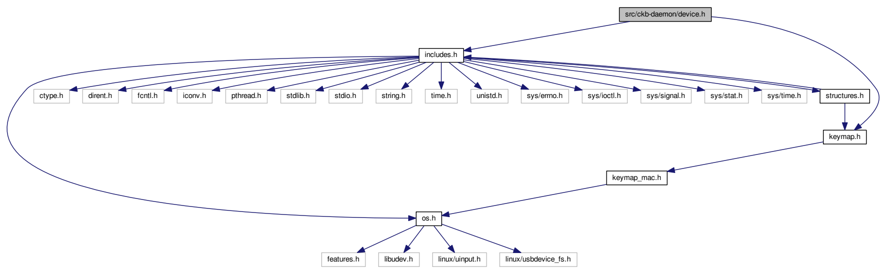
Definition at line 10 of file device.c.

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().

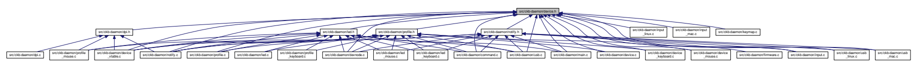
5.4 src/ckb-daemon/device.h File Reference

```
#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))

  setactive() calls via the corresponding kb->vtable either the active() or the idle() function.
  active() is called if the parameter makeactive is true, idle if it is false.
  What function is called effectively is device dependent. Have a look at device_vtable.c for more information.
- #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]

  hwload_mode = 1 means read hardware once. should be enough
- pthread_mutex_t devmutex [9]
- pthread_mutex_t inputmutex [9]

5.4.1 Macro Definition Documentation

5.4.1.1 #define ACT_LIGHT 1

Definition at line 60 of file device.h.

Referenced by setactive_kb().

5.4.1.2 #define ACT_LOCK 8

Definition at line 63 of file device.h.

Referenced by setactive_kb().

5.4.1.3 #define ACT_M1 10

Definition at line 65 of file device.h.

Referenced by setactive_kb().

5.4.1.4 #define ACT_M2 11

Definition at line 66 of file device.h.

Referenced by setactive_kb().

5.4.1.5 #define ACT_M3 12

Definition at line 67 of file device.h.

Referenced by setactive_kb().

5.4.1.6 #define ACT_MR_RING 9

Definition at line 64 of file device.h.

Referenced by setactive_kb().

5.4.1.7 #define ACT_NEXT 3

Definition at line 61 of file device.h.

5.4.1.8 #define ACT_NEXT_NOWRAP 5

Definition at line 62 of file device.h.

5.4.1.9 #define DEV_MAX 9

Definition at line 8 of file device.h.

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

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

Definition at line 18 of file device.h.

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

5.4.1.11 #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().

5.4.1.12 #define IN_CORSAIR 0x40

Definition at line 57 of file device.h.

Referenced by setactive_kb(), and setactive_mouse().

5.4.1.13 #define IN_HID 0x80

Definition at line 56 of file device.h.

Referenced by setactive_kb(), and setactive_mouse().

5.4.1.14 #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().

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

Definition at line 36 of file device.h.

Referenced by _start_dev(), and revertusb().

5.4.2 Function Documentation

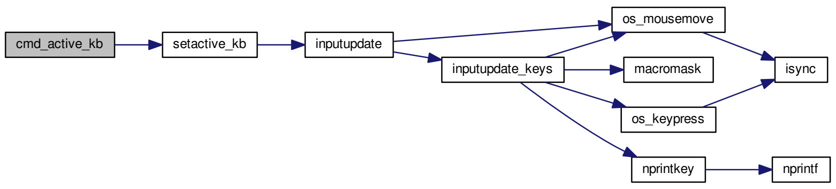
5.4.2.1 int cmd_active_kb (usbdevice * *kb,* usbmode * *dummy1,* int *dummy2,* int *dummy3,* const char * *dummy4*)

Definition at line 112 of file device_keyboard.c.

References setactive_kb().

```
{
    return setactive_kb(kb, 1);
}
```

Here is the call graph for this function:



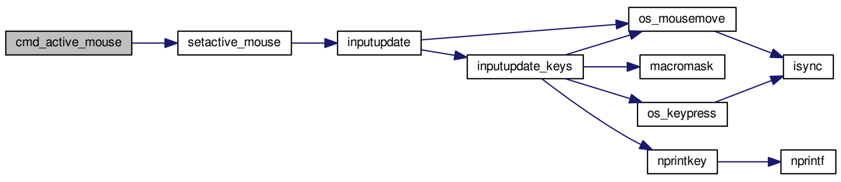
5.4.2.2 int cmd_active_mouse (**usbdevice** ∗ *kb,* **usbmode** ∗ *dummy1,* int *dummy2,* int *dummy3,* const char ∗ *dummy4*)

Definition at line 44 of file device_mouse.c.

References setactive_mouse().

```
{
    return setactive_mouse(kb, 1);
}
```

Here is the call graph for this function:



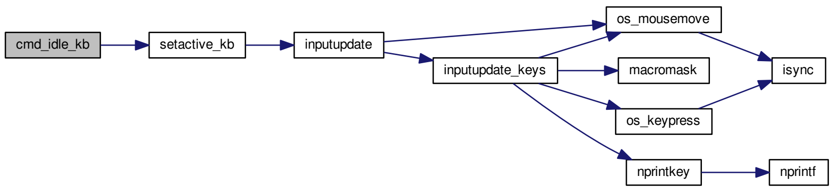
5.4.2.3 int cmd_idle_kb (**usbdevice** ∗ *kb,* **usbmode** ∗ *dummy1,* int *dummy2,* int *dummy3,* const char ∗ *dummy4*)

Definition at line 116 of file device_keyboard.c.

References setactive_kb().

```
{
    return setactive_kb(kb, 0);
}
```

Here is the call graph for this function:



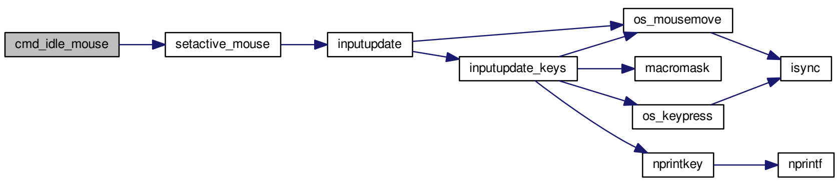
5.4.2.4 int cmd_idle_mouse (usbdevice ∗ *kb,* usbmode ∗ *dummy1,* int *dummy2,* int *dummy3,* const char ∗ *dummy4*)

Definition at line 48 of file device_mouse.c.

References setactive_mouse().

```
                                                                                    {
    return setactive_mouse(kb, 0);
}
```

Here is the call graph for this function:



5.4.2.5 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.

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

5.4.2.6 int setactive_kb (usbdevice ∗ *kb,* int *active*)

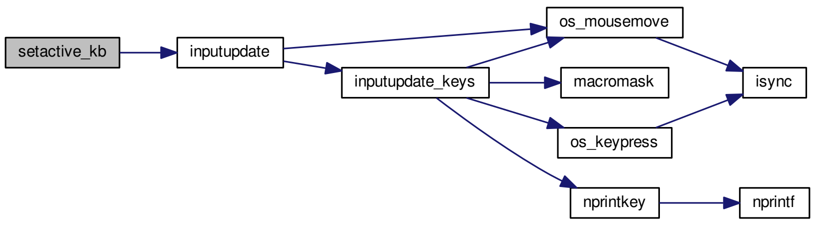
Definition at line 18 of file device_keyboard.c.

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.

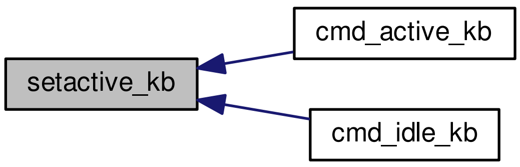
Referenced by cmd_active_kb(), and cmd_idle_kb().

```
                                        {
    if(NEEDS_FW_UPDATE(kb))
        return 0;

    pthread_mutex_lock(imutex(kb));
    kb->active = !!active;
    kb->profile->lastlight.forceupdate = 1;
    // Clear input
    memset(&kb->input.keys, 0, sizeof(kb->input.keys));
    inputupdate(kb);
    pthread_mutex_unlock(imutex(kb));

    uchar msg[3][MSG_SIZE] = {
        { 0x07, 0x04, 0 },                  // Disables or enables HW control for top row
        { 0x07, 0x40, 0 },                  // Selects key input
        { 0x07, 0x05, 2, 0, 0x03, 0x00 }    // Commits key input selection
    };
    if(active){
        // Put the M-keys (K95) as well as the Brightness/Lock keys into software-controlled mode.
        msg[0][2] = 2;
        if(!usbsend(kb, msg[0], 1))
            return -1;
        DELAY_MEDIUM(kb);
        // Set input mode on the keys. They must be grouped into packets of 60 bytes (+ 4 bytes header)
        // Keys are referenced in byte pairs, with the first byte representing the key and the second byte representing the mode.
        for(int key = 0; key < N_KEYS_HW; ){
            int pair;
            for(pair = 0; pair < 30 && key < N_KEYS_HW; pair++, key++){
                // Select both standard and Corsair input. The standard input will be ignored except in BIOS mode.
                uchar action = IN_HID | IN_CORSAIR;
                // Additionally, make MR activate the MR ring (this is disabled for now, may be back later)
                //if(keymap[key].name && !strcmp(keymap[key].name, "mr"))
                //    action |= ACT_MR_RING;
                msg[1][4 + pair * 2] = key;
                msg[1][5 + pair * 2] = action;
            }
            // Byte 2 = pair count (usually 30, less on final message)
            msg[1][2] = pair;
            if(!usbsend(kb, msg[1], 1))
                return -1;
        }
        // Commit new input settings
        if(!usbsend(kb, msg[2], 1))
            return -1;
        DELAY_MEDIUM(kb);
    } else {
        // Set the M-keys back into hardware mode, restore hardware RGB profile. It has to be sent twice for some reason.
        msg[0][2] = 1;
        if(!usbsend(kb, msg[0], 1))
            return -1;
        DELAY_MEDIUM(kb);
        if(!usbsend(kb, msg[0], 1))
            return -1;
        DELAY_MEDIUM(kb);
#ifdef OS_LINUX
        // On OSX the default key mappings are fine. On Linux, the G keys will freeze the keyboard. Set the keyboard entirely to HID input.
        for(int key = 0; key < N_KEYS_HW; ){
            int pair;
            for(pair = 0; pair < 30 && key < N_KEYS_HW; pair++, key++){
                uchar action = IN_HID;
                // Enable hardware actions
                if(keymap[key].name){
                    if(!strcmp(keymap[key].name, "mr"))
                        action = ACT_MR_RING;
                    else if(!strcmp(keymap[key].name, "m1"))
                        action = ACT_M1;
                    else if(!strcmp(keymap[key].name, "m2"))
                        action = ACT_M2;
                    else if(!strcmp(keymap[key].name, "m3"))
                        action = ACT_M3;
                    else if(!strcmp(keymap[key].name, "light"))
                        action = ACT_LIGHT;
                    else if(!strcmp(keymap[key].name, "lock"))
                        action = ACT_LOCK;
                }
                msg[1][4 + pair * 2] = key;
                msg[1][5 + pair * 2] = action;
            }
            // Byte 2 = pair count (usually 30, less on final message)
            msg[1][2] = pair;
            if(!usbsend(kb, msg[1], 1))
                return -1;
        }
        // Commit new input settings
        if(!usbsend(kb, msg[2], 1))
            return -1;
        DELAY_MEDIUM(kb);
#endif
    }
    // Update indicator LEDs if the profile contains settings for them
    kb->vtable->updateindicators(kb, 0);
    return 0;
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.4.2.7 int setactive_mouse (usbdevice ∗ *kb,* int *active*)

Definition at line 9 of file device_mouse.c.

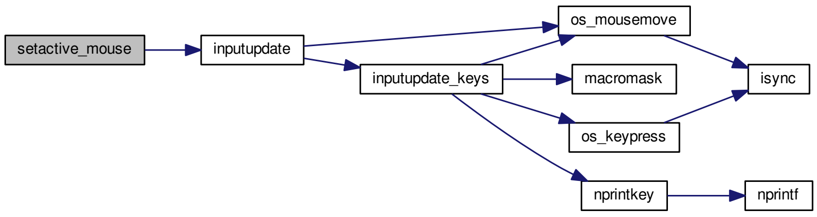
References usbdevice::active, lighting::forceupdate, imutex, IN_CORSAIR, IN_HID, usbdevice::input, inputupdate(), usbinput::keys, usbprofile::lastlight, MSG_SIZE, NEEDS_FW_UPDATE, usbdevice::profile, and usbsend.

Referenced by cmd_active_mouse(), and cmd_idle_mouse().

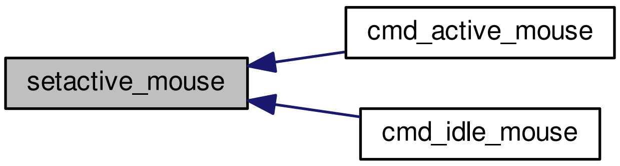
```
                                             {
    if(NEEDS_FW_UPDATE(kb))
        return 0;
    const int keycount = 20;
    uchar msg[2][MSG_SIZE] = {
        { 0x07, 0x04, 0 },                  // Disables or enables HW control for DPI and Sniper button
        { 0x07, 0x40, keycount, 0 },        // Select button input (simliar to the packet sent to
    keyboards, but lacks a commit packet)
    };
    if(active)
        // Put the mouse into SW mode
        msg[0][2] = 2;
    else
        // Restore HW mode
        msg[0][2] = 1;
    pthread_mutex_lock(imutex(kb));
    kb->active = !!active;
    kb->profile->lastlight.forceupdate = 1;
    // Clear input
    memset(&kb->input.keys, 0, sizeof(kb->input.keys));
    inputupdate(kb);
    pthread_mutex_unlock(imutex(kb));
    if(!usbsend(kb, msg[0], 1))
        return -1;
    if(active){
        // Set up key input
        if(!usbsend(kb, msg[1], 1))
```

```
            return -1;
        for(int i = 0; i < keycount; i++){
            msg[1][i * 2 + 4] = i + 1;
            msg[1][i * 2 + 5] = (i < 6 ? IN_HID : IN_CORSAIR);
        }
    }
    return 0;
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.4.2.8 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.

```
                                                {
    switch(index % 3){
    case 0:
        nk95cmd(kb, NK95_M1);
        break;
    case 1:
        nk95cmd(kb, NK95_M2);
        break;
    case 2:
        nk95cmd(kb, NK95_M3);
        break;
    }
}
```

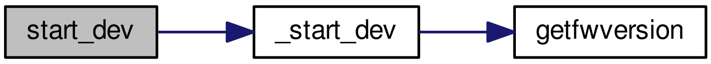
5.4.2.9 int start_dev (usbdevice ∗ *kb,* int *makeactive*)

Definition at line 77 of file device.c.

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

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

Here is the call graph for this function:



5.4.2.10 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.

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

5.4.3 Variable Documentation

5.4.3.1 pthread_mutex_t devmutex[9]

Definition at line 12 of file device.c.

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

5.4.3.2 pthread_mutex_t inputmutex[9]

Definition at line 13 of file device.c.

5.4.3.3 **usbdevice** keyboard[9]

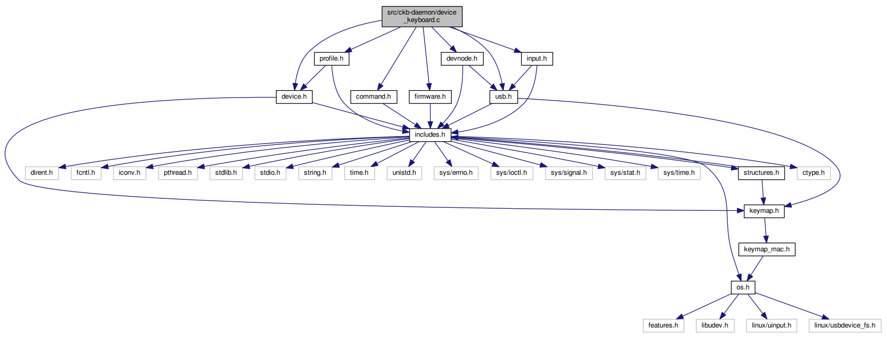
Definition at line 10 of file device.c.

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().

5.5 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)

5.5.1 Function Documentation

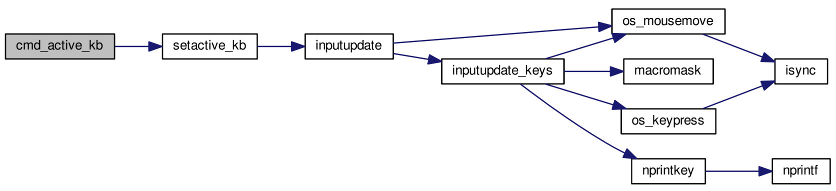
5.5.1.1 int cmd_active_kb (**usbdevice** ∗ ***kb,*** **usbmode** ∗ ***dummy1,*** int ***dummy2,*** int ***dummy3,*** const char ∗ ***dummy4***)

Definition at line 112 of file device_keyboard.c.

References setactive_kb().

```
                                                                                             {
    return setactive_kb(kb, 1);
}
```

Here is the call graph for this function:



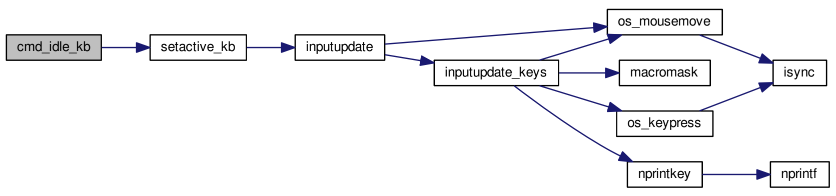
5.5.1.2 int cmd_idle_kb (usbdevice ∗ *kb*, usbmode ∗ *dummy1*, int *dummy2*, int *dummy3*, const char ∗ *dummy4*)

Definition at line 116 of file device_keyboard.c.

References setactive_kb().

```
                                                                                        {
    return setactive_kb(kb, 0);
}
```

Here is the call graph for this function:



5.5.1.3 int setactive_kb (usbdevice ∗ *kb*, int *active*)

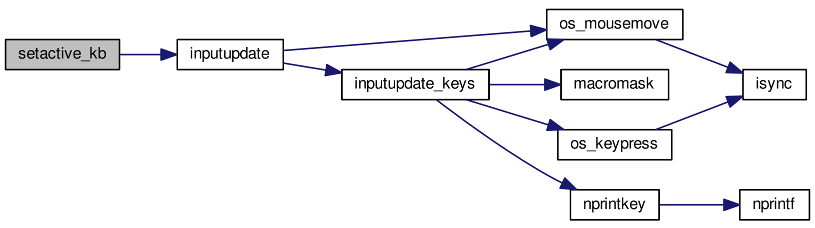
Definition at line 18 of file device_keyboard.c.

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.

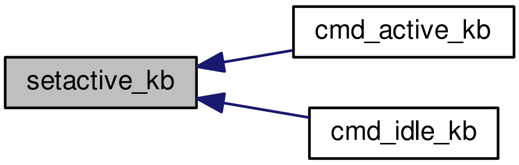
Referenced by cmd_active_kb(), and cmd_idle_kb().

```
                                          {
    if(NEEDS_FW_UPDATE(kb))
        return 0;

    pthread_mutex_lock(imutex(kb));
    kb->active = !!active;
    kb->profile->lastlight.forceupdate = 1;
    // Clear input
    memset(&kb->input.keys, 0, sizeof(kb->input.keys));
    inputupdate(kb);
    pthread_mutex_unlock(imutex(kb));

    uchar msg[3][MSG_SIZE] = {
        { 0x07, 0x04, 0 },                  // Disables or enables HW control for top row
        { 0x07, 0x40, 0 },                  // Selects key input
        { 0x07, 0x05, 2, 0, 0x03, 0x00 }    // Commits key input selection
    };
    if(active){
        // Put the M-keys (K95) as well as the Brightness/Lock keys into software-controlled mode.
        msg[0][2] = 2;
        if(!usbsend(kb, msg[0], 1))
            return -1;
        DELAY_MEDIUM(kb);
        // Set input mode on the keys. They must be grouped into packets of 60 bytes (+ 4 bytes header)
        // Keys are referenced in byte pairs, with the first byte representing the key and the second byte
    representing the mode.
        for(int key = 0; key < N_KEYS_HW; ){
            int pair;
            for(pair = 0; pair < 30 && key < N_KEYS_HW; pair++, key++){
                // Select both standard and Corsair input. The standard input will be ignored except in
    BIOS mode.
                uchar action = IN_HID | IN_CORSAIR;
                // Additionally, make MR activate the MR ring (this is disabled for now, may be back later)
                //if(keymap[key].name && !strcmp(keymap[key].name, "mr"))
                //    action |= ACT_MR_RING;
                msg[1][4 + pair * 2] = key;
                msg[1][5 + pair * 2] = action;
            }
            // Byte 2 = pair count (usually 30, less on final message)
            msg[1][2] = pair;
            if(!usbsend(kb, msg[1], 1))
                return -1;
        }
        // Commit new input settings
        if(!usbsend(kb, msg[2], 1))
            return -1;
        DELAY_MEDIUM(kb);
    } else {
        // Set the M-keys back into hardware mode, restore hardware RGB profile. It has to be sent twice
    for some reason.
        msg[0][2] = 1;
        if(!usbsend(kb, msg[0], 1))
            return -1;
        DELAY_MEDIUM(kb);
        if(!usbsend(kb, msg[0], 1))
            return -1;
        DELAY_MEDIUM(kb);
#ifdef OS_LINUX
        // On OSX the default key mappings are fine. On Linux, the G keys will freeze the keyboard. Set the
    keyboard entirely to HID input.
        for(int key = 0; key < N_KEYS_HW; ){
            int pair;
            for(pair = 0; pair < 30 && key < N_KEYS_HW; pair++, key++){
                uchar action = IN_HID;
                // Enable hardware actions
                if(keymap[key].name){
                    if(!strcmp(keymap[key].name, "mr"))
                        action = ACT_MR_RING;
                    else if(!strcmp(keymap[key].name, "m1"))
                        action = ACT_M1;
                    else if(!strcmp(keymap[key].name, "m2"))
                        action = ACT_M2;
                    else if(!strcmp(keymap[key].name, "m3"))
                        action = ACT_M3;
                    else if(!strcmp(keymap[key].name, "light"))
                        action = ACT_LIGHT;
                    else if(!strcmp(keymap[key].name, "lock"))
                        action = ACT_LOCK;
                }
                msg[1][4 + pair * 2] = key;
                msg[1][5 + pair * 2] = action;
            }
            // Byte 2 = pair count (usually 30, less on final message)
            msg[1][2] = pair;
            if(!usbsend(kb, msg[1], 1))
                return -1;
        }
        // Commit new input settings
        if(!usbsend(kb, msg[2], 1))
            return -1;
        DELAY_MEDIUM(kb);
#endif
    }
    // Update indicator LEDs if the profile contains settings for them
    kb->vtable->updateindicators(kb, 0);
    return 0;
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.5.1.4 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.

```
                                              {
    switch(index % 3){
    case 0:
        nk95cmd(kb, NK95_M1);
        break;
    case 1:
        nk95cmd(kb, NK95_M2);
        break;
    case 2:
        nk95cmd(kb, NK95_M3);
        break;
    }
}
```

5.5.1.5 int start_kb_nrgb (usbdevice ∗ *kb,* int *makeactive*)

Definition at line 9 of file device_keyboard.c.

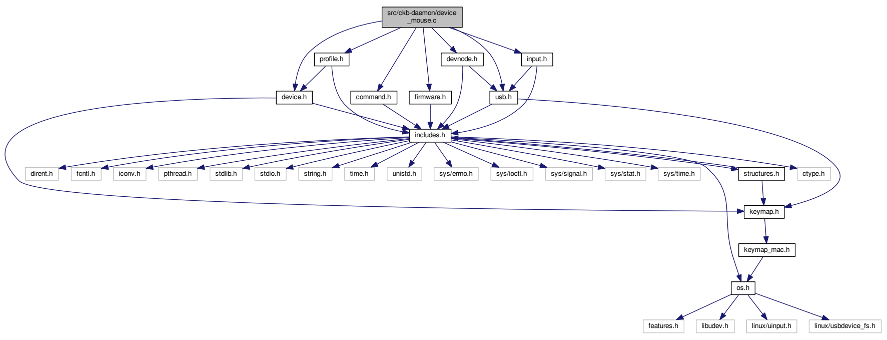
References usbdevice::active, NK95_HWOFF, nk95cmd, and usbdevice::pollrate.

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

5.6 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)

5.6.1 Function Documentation

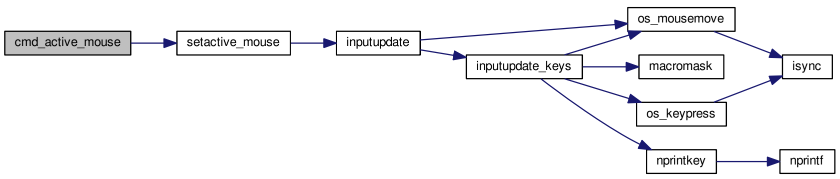
5.6.1.1 int cmd_active_mouse (**usbdevice** ∗ ***kb,*** **usbmode** ∗ ***dummy1,*** int ***dummy2,*** int ***dummy3,*** const char ∗ ***dummy4***)

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:



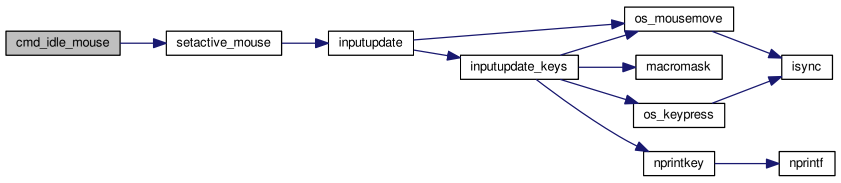
5.6.1.2 int cmd_idle_mouse (**usbdevice** ∗ ***kb,*** **usbmode** ∗ ***dummy1,*** int ***dummy2,*** int ***dummy3,*** const char ∗ ***dummy4***)

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:



5.6.1.3 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.

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

5.6.1.4 int setactive_mouse (usbdevice ∗ *kb*, int *active*)

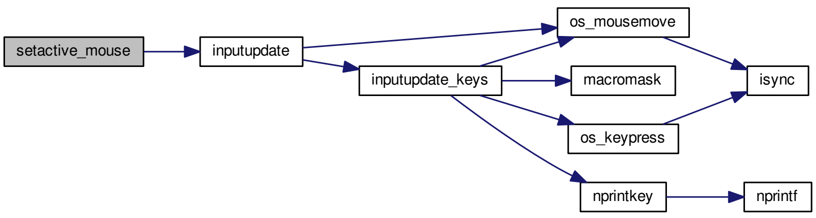
Definition at line 9 of file device_mouse.c.

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.

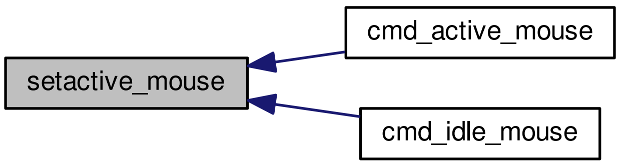
Referenced by cmd_active_mouse(), and cmd_idle_mouse().

```
                                            {
    if(NEEDS_FW_UPDATE(kb))
        return 0;
    const int keycount = 20;
    uchar msg[2][MSG_SIZE] = {
        { 0x07, 0x04, 0 },                 // Disables or enables HW control for DPI and Sniper button
        { 0x07, 0x40, keycount, 0 },       // Select button input (simlilar to the packet sent to
    keyboards, but lacks a commit packet)
    };
    if(active)
        // Put the mouse into SW mode
        msg[0][2] = 2;
    else
        // Restore HW mode
        msg[0][2] = 1;
    pthread_mutex_lock(imutex(kb));
    kb->active = !!active;
    kb->profile->lastlight.forceupdate = 1;
    // Clear input
    memset(&kb->input.keys, 0, sizeof(kb->input.keys));
    inputupdate(kb);
    pthread_mutex_unlock(imutex(kb));
    if(!usbsend(kb, msg[0], 1))
        return -1;
    if(active){
        // Set up key input
        if(!usbsend(kb, msg[1], 1))
            return -1;
        for(int i = 0; i < keycount; i++){
            msg[1][i * 2 + 4] = i + 1;
            msg[1][i * 2 + 5] = (i < 6 ? IN_HID : IN_CORSAIR);
        }
    }
    return 0;
}
```

Here is the call graph for this function:



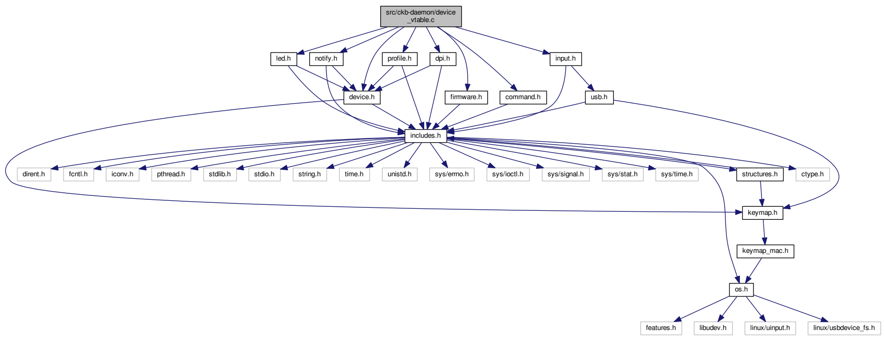
Here is the caller graph for this function:



5.7 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

5.7.1 Function Documentation

5.7.1.1 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 }
```

5.7.1.2 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 }
```

5.7.1.3 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.

```
                                                                                                          {
}
```

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

Definition at line 23 of file device_vtable.c.

```
                                                    {
    return 0;
}
```

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

Definition at line 21 of file device_vtable.c.

```
                                                      {
}
```

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

Definition at line 18 of file device_vtable.c.

```
                                            {
    return 0;
}
```

5.7.2 Variable Documentation

5.7.2.1 const devcmd vtable_keyboard

Definition at line 28 of file device_vtable.c.

Referenced by get_vtable().

5.7.2.2 const devcmd vtable_keyboard_nonrgb

Definition at line 75 of file device_vtable.c.

Referenced by get_vtable().

5.7.2.3 const devcmd vtable_mouse

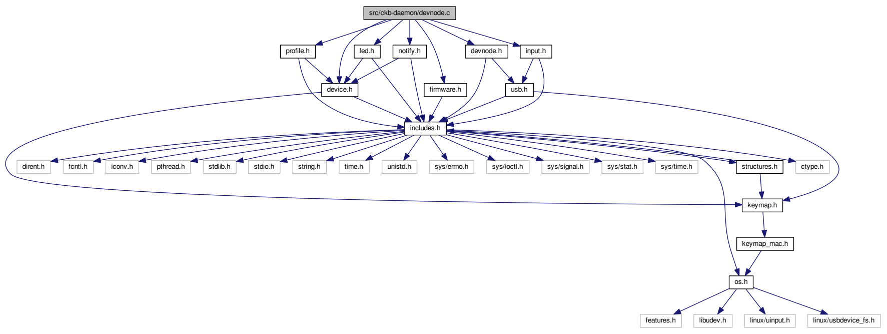
Definition at line 122 of file device_vtable.c.

Referenced by get_vtable().

5.8 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 ()

  _updateconnected Update the list of connected devices.
- void updateconnected ()

  Update the list of connected devices.
- int _mknotifynode (usbdevice ∗kb, int notify)
- int mknotifynode (usbdevice ∗kb, int notify)

  Creates a notification node for the specified keyboard.
- int _rmnotifynode (usbdevice ∗kb, int notify)
- int rmnotifynode (usbdevice ∗kb, int notify)

  Removes a notification node for the specified keyboard.
- static int _mkdevpath (usbdevice ∗kb)
- int mkdevpath (usbdevice ∗kb)

  Create a dev path for the keyboard at index. Returns 0 on success.
- int rmdevpath (usbdevice ∗kb)

  Remove the dev path for the keyboard at index. Returns 0 on success.
- int mkfwnode (usbdevice ∗kb)

Writes a keyboard's firmware version and poll rate to its device node.

- 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

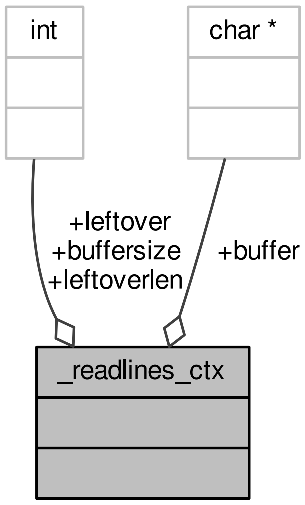
  Group ID for the control nodes. -1 to give read/write access to everybody.

5.8.1 Data Structure Documentation

5.8.1.1 struct _readlines_ctx

Definition at line 335 of file devnode.c.

Collaboration diagram for _readlines_ctx:



Data Fields

char ∗	buffer	
int	buffersize	
int	leftover	
int	leftoverlen	

5.8.2 Macro Definition Documentation

5.8.2.1 #define MAX_BUFFER (1024 ∗ 1024 - 1)

Definition at line 334 of file devnode.c.

Referenced by readlines().

5.8.2.2 #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().

5.8.3 Function Documentation

5.8.3.1 static int _mkdevpath (usbdevice ∗ *kb*) [static]

Definition at line 136 of file devnode.c.

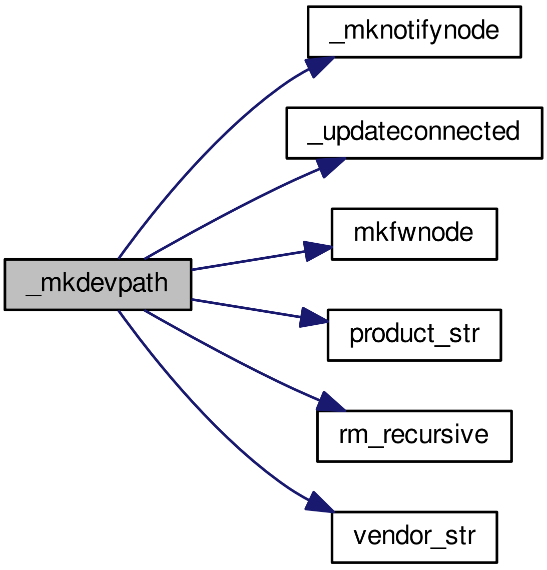
References _mknotifynode(), _updateconnected(), ckb_err, ckb_warn, devpath, FEAT_ADJRATE, FEAT_BIND, FEAT_FWUPDATE, FEAT_FWVERSION, FEAT_MONOCHROME, FEAT_NOTIFY, FEAT_POLLRATE, FEAT_-RGB, gid, HAS_FEATURES, INDEX_OF, usbdevice::infifo, keyboard, mkfwnode(), usbdevice::name, usbdevice-::product, product_str(),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

Referenced by mkdevpath().

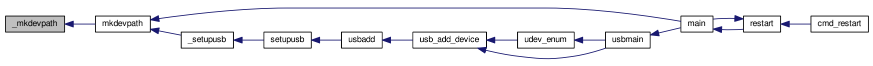
```
136                                   {
137     int index = INDEX_OF(kb, keyboard);
138     // Create the control path
139     char path[strlen(devpath) + 2];
140     snprintf(path, sizeof(path), "%s%d", devpath, index);
141     if(rm_recursive(path) != 0 && errno != ENOENT){
142         ckb_err("Unable to delete %s: %s\n", path, strerror(errno));
143         return -1;
144     }
145     if(mkdir(path, S_READDIR) != 0){
146         ckb_err("Unable to create %s: %s\n", path, strerror(errno));
147         rm_recursive(path);
148         return -1;
149     }
150     if(gid >= 0)
151         chown(path, 0, gid);
152
153     if(kb == keyboard + 0){
154         // Root keyboard: write a list of devices
155         _updateconnected();
156         // Write version number
157         char vpath[sizeof(path) + 8];
158         snprintf(vpath, sizeof(vpath), "%s/version", path);
159         FILE* vfile = fopen(vpath, "w");
160         if(vfile){
161             fprintf(vfile, "%s\n", CKB_VERSION_STR);
162             fclose(vfile);
163             chmod(vpath, S_GID_READ);
164             if(gid >= 0)
165                 chown(vpath, 0, gid);
166         } else {
167             ckb_warn("Unable to create %s: %s\n", vpath, strerror(errno));
168             remove(vpath);
169         }
170         // Write PID
171         char ppath[sizeof(path) + 4];
172         snprintf(ppath, sizeof(ppath), "%s/pid", path);
173         FILE* pfile = fopen(ppath, "w");
174         if(pfile){
175             fprintf(pfile, "%u\n", getpid());
176             fclose(pfile);
177             chmod(ppath, S_READ);
178             if(gid >= 0)
179                 chown(vpath, 0, gid);
180         } else {
181             ckb_warn("Unable to create %s: %s\n", ppath, strerror(errno));
182             remove(ppath);
183         }
184     } else {
185         // Create command FIFO
186         char inpath[sizeof(path) + 4];
187         snprintf(inpath, sizeof(inpath), "%s/cmd", path);
188         if(mkfifo(inpath, gid >= 0 ? S_CUSTOM : S_READWRITE) != 0
189                 // Open the node in RDWR mode because RDONLY will lock the thread
```

```
                || (kb->infifo = open(inpath, O_RDWR) + 1) == 0){
            // Add one to the FD because 0 is a valid descriptor, but ckb uses 0 for uninitialized devices
            ckb_err("Unable to create %s: %s\n", inpath, strerror(errno));
            rm_recursive(path);
            kb->infifo = 0;
            return -1;
        }
        if(gid >= 0)
            fchown(kb->infifo - 1, 0, gid);

        // Create notification FIFO
        _mknotifynode(kb, 0);

        // Write the model and serial to files
        char mpath[sizeof(path) + 6], spath[sizeof(path) + 7];
        snprintf(mpath, sizeof(mpath), "%s/model", path);
        snprintf(spath, sizeof(spath), "%s/serial", path);
        FILE* mfile = fopen(mpath, "w");
        if(mfile){
            fputs(kb->name, mfile);
            fputc('\n', mfile);
            fclose(mfile);
            chmod(mpath, S_GID_READ);
            if(gid >= 0)
                chown(mpath, 0, gid);
        } else {
            ckb_warn("Unable to create %s: %s\n", mpath, strerror(errno));
            remove(mpath);
        }
        FILE* sfile = fopen(spath, "w");
        if(sfile){
            fputs(kb->serial, sfile);
            fputc('\n', sfile);
            fclose(sfile);
            chmod(spath, S_GID_READ);
            if(gid >= 0)
                chown(spath, 0, gid);
        } else {
            ckb_warn("Unable to create %s: %s\n", spath, strerror(errno));
            remove(spath);
        }
        // Write the keyboard's features
        char fpath[sizeof(path) + 9];
        snprintf(fpath, sizeof(fpath), "%s/features", path);
        FILE* ffile = fopen(fpath, "w");
        if(ffile){
            fprintf(ffile, "%s %s", vendor_str(kb->vendor),
    product_str(kb->product));
            if(HAS_FEATURES(kb, FEAT_MONOCHROME))
                fputs(" monochrome", ffile);
            if(HAS_FEATURES(kb, FEAT_RGB))
                fputs(" rgb", ffile);
            if(HAS_FEATURES(kb, FEAT_POLLRATE))
                fputs(" pollrate", ffile);
            if(HAS_FEATURES(kb, FEAT_ADJRATE))
                fputs(" adjrate", ffile);
            if(HAS_FEATURES(kb, FEAT_BIND))
                fputs(" bind", ffile);
            if(HAS_FEATURES(kb, FEAT_NOTIFY))
                fputs(" notify", ffile);
            if(HAS_FEATURES(kb, FEAT_FWVERSION))
                fputs(" fwversion", ffile);
            if(HAS_FEATURES(kb, FEAT_FWUPDATE))
                fputs(" fwupdate", ffile);
            fputc('\n', ffile);
            fclose(ffile);
            chmod(fpath, S_GID_READ);
            if(gid >= 0)
                chown(fpath, 0, gid);
        } else {
            ckb_warn("Unable to create %s: %s\n", fpath, strerror(errno));
            remove(fpath);
        }
        // Write firmware version and poll rate
        mkfwnode(kb);
    }
    return 0;
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.8.3.2 int _mknotifynode (**usbdevice** ∗ *kb,* int *notify*)

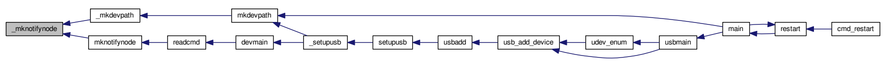
Definition at line 87 of file devnode.c.

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

Referenced by _mkdevpath(), and mknotifynode().

```
87                                             {
88      if(notify < 0 || notify >= OUTFIFO_MAX)
89          return -1;
90      if(kb->outfifo[notify] != 0)
91          return 0;
92      // Create the notification node
93      int index = INDEX_OF(kb, keyboard);
94      char outpath[strlen(devpath) + 10];
95      snprintf(outpath, sizeof(outpath), "%s%d/notify%d", devpath, index, notify);
96      if(mkfifo(outpath, S_GID_READ) != 0 || (kb->outfifo[notify] = open(outpath, O_RDWR |
      O_NONBLOCK) + 1) == 0){
97          // Add one to the FD because 0 is a valid descriptor, but ckb uses 0 for uninitialized devices
98          ckb_warn("Unable to create %s: %s\n", outpath, strerror(errno));
99          kb->outfifo[notify] = 0;
100          remove(outpath);
101          return -1;
102      }
103      if(gid >= 0)
104          fchown(kb->outfifo[notify] - 1, 0, gid);
105      return 0;
106 }
```

Here is the caller graph for this function:



5.8.3.3 int _rmnotifynode (usbdevice ∗ *kb*, int *notify*)

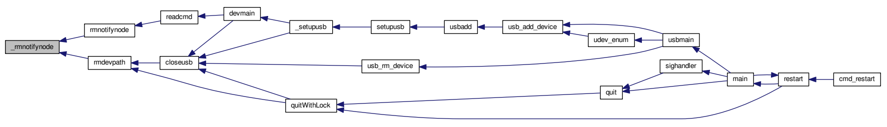
Definition at line 115 of file devnode.c.

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

Referenced by rmdevpath(), and rmnotifynode().

```
115                                          {
116     if(notify < 0 || notify >= OUTFIFO_MAX || !kb->outfifo[notify])
117         return -1;
118     int index = INDEX_OF(kb, keyboard);
119     char outpath[strlen(devpath) + 10];
120     snprintf(outpath, sizeof(outpath), "%s%d/notify%d", devpath, index, notify);
121     // Close FIFO
122     close(kb->outfifo[notify] - 1);
123     kb->outfifo[notify] = 0;
124     // Delete node
125     int res = remove(outpath);
126     return res;
127 }
```

Here is the caller graph for this function:



5.8.3.4 void _updateconnected ( )

<devicepath> normally is /dev/input/ckb or /input/ckb.

Open the normal file under <devicepath>0/connected for writing. For each device connected, print its devicepath+number, the serial number of the usb device and the usb name of the device connected to that usb interface.

eg:

/dev/input/ckb1 0F022014ABABABABABABABABABABA999 Corsair K95 RGB Gaming Keyboard

/dev/input/ckb2 0D02303DBACBACBACBACBACBACBACBAC998 Corsair M65 RGB Gaming Mouse

Set the file ownership to root. If the glob var gid is explicitly set to something different from -1 (the initial value), set file permission to 640, else to 644. This is used if you start the daemon with –gid=<GID> Parameter.

Because several independent threads may call updateconnected(), protect that procedure with locking/unlocking of **devmutex**.

Definition at line 55 of file devnode.c.

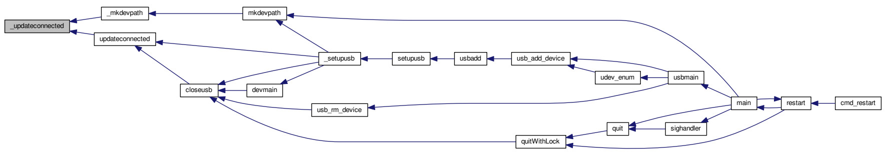
References ckb_warn, DEV_MAX, devmutex, devpath, gid, IS_CONNECTED, keyboard, and S_GID_READ.

Referenced by _mkdevpath(), and updateconnected().

```
55                           {
```

```
56     pthread_mutex_lock(devmutex);
57     char cpath[strlen(devpath) + 12];
58     snprintf(cpath, sizeof(cpath), "%s0/connected", devpath);
59     FILE* cfile = fopen(cpath, "w");
60     if(!cfile){
61         ckb_warn("Unable to update %s: %s\n", cpath, strerror(errno));
62         pthread_mutex_unlock(devmutex);
63         return;
64     }
65     int written = 0;
66     for(int i = 1; i < DEV_MAX; i++){
67         if(IS_CONNECTED(keyboard + i)){
68             written = 1;
69             fprintf(cfile, "%s%d %s %s\n", devpath, i, keyboard[i].serial,
      keyboard[i].name);
70         }
71     }
72     if(!written)
73         fputc('\n', cfile);
74     fclose(cfile);
75     chmod(cpath, S_GID_READ);
76     if(gid >= 0)
77         chown(cpath, 0, gid);
78     pthread_mutex_unlock(devmutex);
79 }
```

Here is the caller graph for this function:



5.8.3.5 int mkdevpath (**usbdevice** ∗ ***kb***)

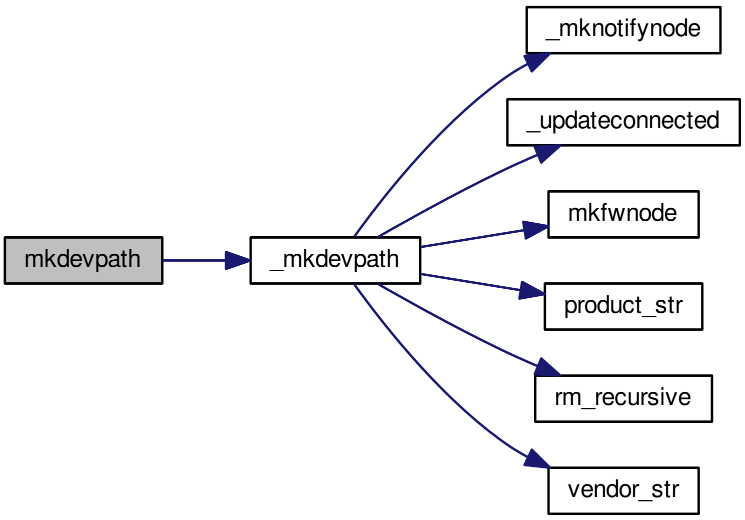
Definition at line 268 of file devnode.c.

References _mkdevpath(), euid_guard_start, and euid_guard_stop.

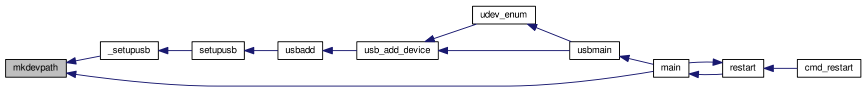
Referenced by _setupusb(), and main().

```
268                             {
269     euid_guard_start;
270     int res = _mkdevpath(kb);
271     euid_guard_stop;
272     return res;
273 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.8.3.6 int mkfwnode (**usbdevice** ∗ *kb*)

Definition at line 299 of file devnode.c.

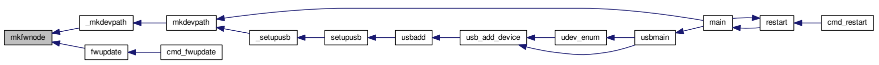
References ckb_warn, devpath, usbdevice::fwversion, gid, INDEX_OF, keyboard, usbdevice::pollrate, and S_GID_-READ.

Referenced by _mkdevpath(), and fwupdate().

```
299                              {
300     int index = INDEX_OF(kb, keyboard);
301     char fwpath[strlen(devpath) + 12];
302     snprintf(fwpath, sizeof(fwpath), "%s%d/fwversion", devpath, index);
303     FILE* fwfile = fopen(fwpath, "w");
304     if(fwfile){
305         fprintf(fwfile, "%04x", kb->fwversion);
306         fputc('\n', fwfile);
307         fclose(fwfile);
308         chmod(fwpath, S_GID_READ);
309         if(gid >= 0)
310             chown(fwpath, 0, gid);
311     } else {
312         ckb_warn("Unable to create %s: %s\n", fwpath, strerror(errno));
313         remove(fwpath);
314         return -1;
315     }
316     char ppath[strlen(devpath) + 11];
317     snprintf(ppath, sizeof(ppath), "%s%d/pollrate", devpath, index);
318     FILE* pfile = fopen(ppath, "w");
```

```
    if(pfile){
        fprintf(pfile, "%d ms", kb->pollrate);
        fputc('\n', pfile);
        fclose(pfile);
        chmod(ppath, S_GID_READ);
        if(gid >= 0)
            chown(ppath, 0, gid);
    } else {
        ckb_warn("Unable to create %s: %s\n", fwpath, strerror(errno));
        remove(ppath);
        return -2;
    }
    return 0;
}
```

Here is the caller graph for this function:



5.8.3.7 int mknotifynode (usbdevice ∗ *kb,* int *notify*)

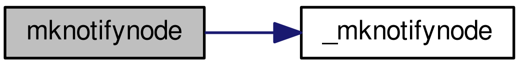
Definition at line 108 of file devnode.c.

References _mknotifynode(), euid_guard_start, and euid_guard_stop.

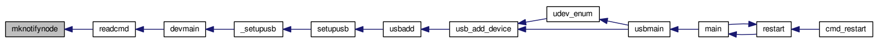
Referenced by readcmd().

```
                                             {
    euid_guard_start;
    int res = _mknotifynode(kb, notify);
    euid_guard_stop;
    return res;
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



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

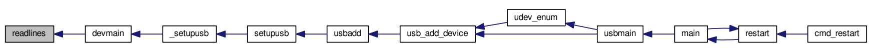
Definition at line 353 of file devnode.c.

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

Referenced by devmain().

```
353                                                                 {
354     // Move any data left over from a previous read to the start of the buffer
355     char* buffer = ctx->buffer;
356     int buffersize = ctx->buffersize;
357     int leftover = ctx->leftover, leftoverlen = ctx->leftoverlen;
358     memcpy(buffer, buffer + leftover, leftoverlen);
359     // Read data from the file
360     ssize_t length = read(fd, buffer + leftoverlen, buffersize - leftoverlen);
361     length = (length < 0 ? 0 : length) + leftoverlen;
362     leftover = ctx->leftover = leftoverlen = ctx->leftoverlen = 0;
363     if(length <= 0){
364         *input = 0;
365         return 0;
366     }
367     // Continue buffering until all available input is read or there's no room left
368     while(length == buffersize){
369         if(buffersize == MAX_BUFFER)
370             break;
371         int oldsize = buffersize;
372         buffersize += 4096;
373         ctx->buffersize = buffersize;
374         buffer = ctx->buffer = realloc(buffer, buffersize + 1);
375         ssize_t length2 = read(fd, buffer + oldsize, buffersize - oldsize);
376         if(length2 <= 0)
377             break;
378         length += length2;
379     }
380     buffer[length] = 0;
381     // Input should be issued one line at a time and should end with a newline.
382     char* lastline = memrchr(buffer, '\n', length);
383     if(lastline == buffer + length - 1){
384         // If the buffer ends in a newline, process the whole string
385         *input = buffer;
386         return length;
387     } else if(lastline){
388         // Otherwise, chop off the last line but process everything else
389         *lastline = 0;
390         leftover = ctx->leftover = lastline + 1 - buffer;
391         leftoverlen = ctx->leftoverlen = length - leftover;
392         *input = buffer;
393         return leftover - 1;
394     } else {
395         // If a newline wasn't found at all, process the whole buffer next time
396         *input = 0;
397         if(length == MAX_BUFFER){
398             // Unless the buffer is completely full, in which case discard it
399             ckb_warn("Too much input (1MB). Dropping.\n");
400             return 0;
401         }
402         leftoverlen = ctx->leftoverlen = length;
403         return 0;
404     }
405 }
```

Here is the caller graph for this function:



5.8.3.9 void readlines_ctx_free (readlines_ctx *ctx*)

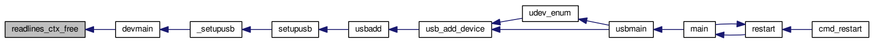
Definition at line 348 of file devnode.c.

References _readlines_ctx::buffer.

Referenced by devmain().

```
348                                              {
    free(ctx->buffer);
    free(ctx);
}
```

Here is the caller graph for this function:



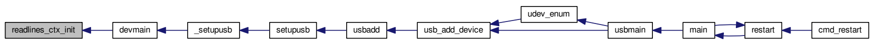
5.8.3.10 void readlines_ctx_init (readlines_ctx ∗ *ctx*)

Definition at line 341 of file devnode.c.

Referenced by devmain().

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

Here is the caller graph for this function:



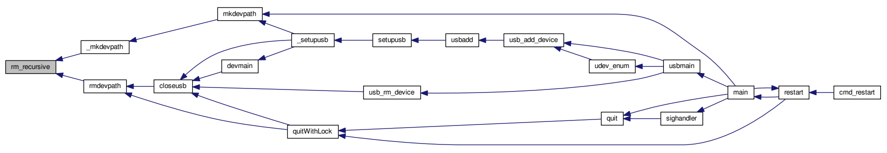
5.8.3.11 int rm_recursive (const char ∗ *path*)

Definition at line 19 of file devnode.c.

Referenced by _mkdevpath(), and rmdevpath().

```
                                   {
    DIR* dir = opendir(path);
    if(!dir)
        return remove(path);
    struct dirent* file;
    while((file = readdir(dir)))
    {
        if(!strcmp(file->d_name, ".") || !strcmp(file->d_name, ".."))
            continue;
        char path2[FILENAME_MAX];
        snprintf(path2, FILENAME_MAX, "%s/%s", path, file->d_name);
        int stat = rm_recursive(path2);
        if(stat != 0)
            return stat;
    }
    closedir(dir);
    return remove(path);
}
```

Here is the caller graph for this function:



5.8.3.12 int rmdevpath (**usbdevice** ∗ *kb*)

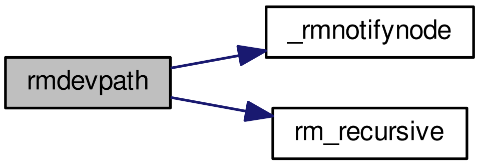
Definition at line 275 of file devnode.c.

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

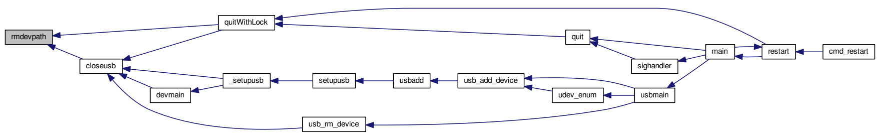
Referenced by closeusb(), and quitWithLock().

```
275                             {
276     euid_guard_start;
277     int index = INDEX_OF(kb, keyboard);
278     if(kb->infifo != 0){
279 #ifdef OS_LINUX
280         write(kb->infifo - 1, "\n", 1); // hack to prevent the FIFO thread from perma-blocking
281 #endif
282         close(kb->infifo - 1);
283         kb->infifo = 0;
284     }
285     for(int i = 0; i < OUTFIFO_MAX; i++)
286         _rmnotifynode(kb, i);
287     char path[strlen(devpath) + 2];
288     snprintf(path, sizeof(path), "%s%d", devpath, index);
289     if(rm_recursive(path) != 0 && errno != ENOENT){
290         ckb_warn("Unable to delete %s: %s\n", path, strerror(errno));
291         euid_guard_stop;
292         return -1;
293     }
294     ckb_info("Removed device path %s\n", path);
295     euid_guard_stop;
296     return 0;
297 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.8.3.13 int rmnotifynode (**usbdevice** ∗ *kb,* int *notify*)

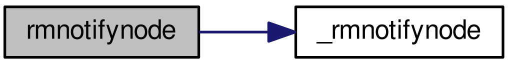
Definition at line 129 of file devnode.c.

References _rmnotifynode(), euid_guard_start, and euid_guard_stop.

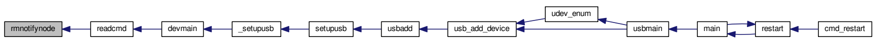
Referenced by readcmd().

```
129                                          {
130     euid_guard_start;
131     int res = _rmnotifynode(kb, notify);
132     euid_guard_stop;
133     return res;
134 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.8.3.14 void updateconnected ( )

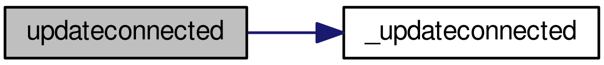
Definition at line 81 of file devnode.c.

References _updateconnected(), euid_guard_start, and euid_guard_stop.

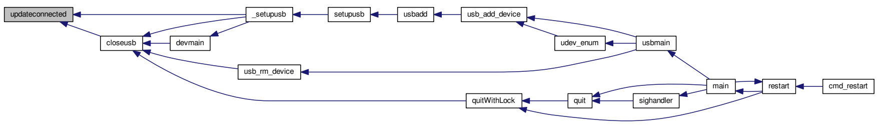
Referenced by _setupusb(), and closeusb().

```
81                        {
82     euid_guard_start;
83     _updateconnected();
84     euid_guard_stop;
85 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.8.4 Variable Documentation

5.8.4.1 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().

5.8.4.2 long gid = -1

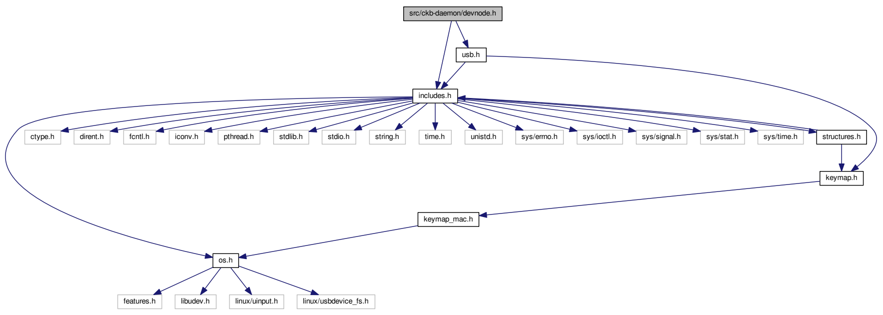
Definition at line 16 of file devnode.c.

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

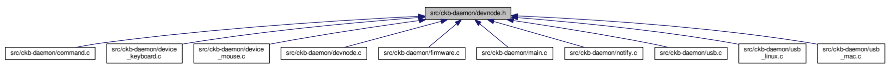
5.9 src/ckb-daemon/devnode.h File Reference

```
#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

  Custom readline is needed for FIFOs. fopen()/getline() will die if the data is sent in too fast.

Functions

- void updateconnected ()

  Update the list of connected devices.
- int mkdevpath (usbdevice ∗kb)

  Create a dev path for the keyboard at index. Returns 0 on success.
- int rmdevpath (usbdevice ∗kb)

  Remove the dev path for the keyboard at index. Returns 0 on success.
- int mknotifynode (usbdevice ∗kb, int notify)

  Creates a notification node for the specified keyboard.
- int rmnotifynode (usbdevice ∗kb, int notify)

  Removes a notification node for the specified keyboard.
- int mkfwnode (usbdevice ∗kb)

  Writes a keyboard's firmware version and poll rate to its device node.
- 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

  Device path base ("/dev/input/ckb" or "/var/run/ckb")
- long gid

  Group ID for the control nodes. -1 to give read/write access to everybody.

5.9.1 Macro Definition Documentation

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

Definition at line 17 of file devnode.h.

Referenced by _mkdevpath().

5.9.1.2 #define S_CUSTOM_R (S_IRUSR | S_IWUSR | S_IRGRP)

Definition at line 18 of file devnode.h.

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

Definition at line 15 of file devnode.h.

Referenced by _mkdevpath().

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

Definition at line 14 of file devnode.h.

Referenced by _mkdevpath().

5.9.1.5 #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().

5.9.2 Typedef Documentation

5.9.2.1 typedef struct _readlines_ctx∗ readlines_ctx

Definition at line 39 of file devnode.h.

5.9.3 Function Documentation

5.9.3.1 int mkdevpath (usbdevice ∗ *kb*)

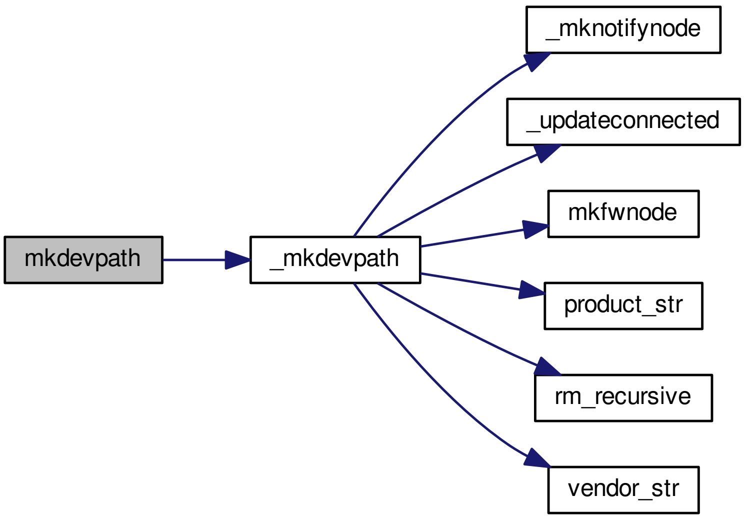
Definition at line 268 of file devnode.c.

References _mkdevpath(), euid_guard_start, and euid_guard_stop.

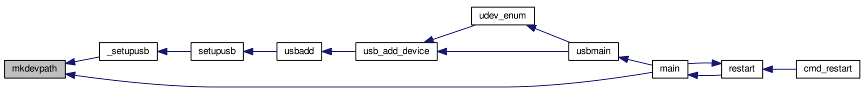
Referenced by _setupusb(), and main().

```
                             {
    euid_guard_start;
    int res = _mkdevpath(kb);
    euid_guard_stop;
    return res;
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.9.3.2 int mkfwnode (**usbdevice** ∗ *kb*)

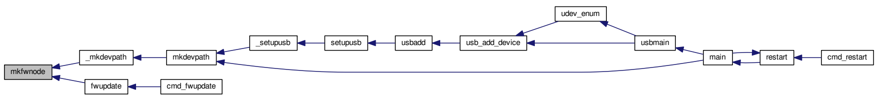
Definition at line 299 of file devnode.c.

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

Referenced by _mkdevpath(), and fwupdate().

```
299                           {
300     int index = INDEX_OF(kb, keyboard);
301     char fwpath[strlen(devpath) + 12];
302     snprintf(fwpath, sizeof(fwpath), "%s%d/fwversion", devpath, index);
303     FILE* fwfile = fopen(fwpath, "w");
304     if(fwfile){
305         fprintf(fwfile, "%04x", kb->fwversion);
306         fputc('\n', fwfile);
307         fclose(fwfile);
308         chmod(fwpath, S_GID_READ);
309         if(gid >= 0)
310             chown(fwpath, 0, gid);
311     } else {
312         ckb_warn("Unable to create %s: %s\n", fwpath, strerror(errno));
313         remove(fwpath);
314         return -1;
315     }
316     char ppath[strlen(devpath) + 11];
317     snprintf(ppath, sizeof(ppath), "%s%d/pollrate", devpath, index);
318     FILE* pfile = fopen(ppath, "w");
319     if(pfile){
320         fprintf(pfile, "%d ms", kb->pollrate);
321         fputc('\n', pfile);
322         fclose(pfile);
323         chmod(ppath, S_GID_READ);
324         if(gid >= 0)
325             chown(ppath, 0, gid);
326     } else {
327         ckb_warn("Unable to create %s: %s\n", fwpath, strerror(errno));
328         remove(ppath);
329         return -2;
330     }
331     return 0;
332 }
```

Here is the caller graph for this function:



5.9.3.3 int mknotifynode (**usbdevice** ∗ *kb,* int *notify*)

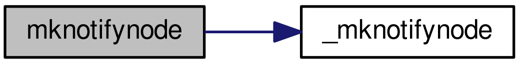
Definition at line 108 of file devnode.c.

References _mknotifynode(), euid_guard_start, and euid_guard_stop.

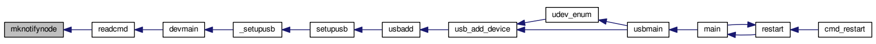
Referenced by readcmd().

```
108                                         {
109     euid_guard_start;
110     int res = _mknotifynode(kb, notify);
111     euid_guard_stop;
112     return res;
113 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



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

Definition at line 353 of file devnode.c.

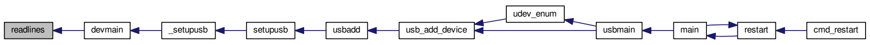
References _readlines_ctx::buffer, _readlines_ctx::buffersize, ckb_warn, _readlines_ctx::leftover, _readlines_ctx::leftoverlen, and MAX_BUFFER.

Referenced by devmain().

```
                                                                    {
    // Move any data left over from a previous read to the start of the buffer
    char* buffer = ctx->buffer;
    int buffersize = ctx->buffersize;
    int leftover = ctx->leftover, leftoverlen = ctx->leftoverlen;
    memcpy(buffer, buffer + leftover, leftoverlen);
    // Read data from the file
    ssize_t length = read(fd, buffer + leftoverlen, buffersize - leftoverlen);
    length = (length < 0 ? 0 : length) + leftoverlen;
    leftover = ctx->leftover = leftoverlen = ctx->leftoverlen = 0;
    if(length <= 0){
        *input = 0;
        return 0;
    }
    // Continue buffering until all available input is read or there's no room left
    while(length == buffersize){
        if(buffersize == MAX_BUFFER)
            break;
        int oldsize = buffersize;
        buffersize += 4096;
        ctx->buffersize = buffersize;
        buffer = ctx->buffer = realloc(buffer, buffersize + 1);
        ssize_t length2 = read(fd, buffer + oldsize, buffersize - oldsize);
        if(length2 <= 0)
            break;
        length += length2;
    }
    buffer[length] = 0;
    // Input should be issued one line at a time and should end with a newline.
    char* lastline = memrchr(buffer, '\n', length);
    if(lastline == buffer + length - 1){
        // If the buffer ends in a newline, process the whole string
        *input = buffer;
        return length;
    } else if(lastline){
        // Otherwise, chop off the last line but process everything else
        *lastline = 0;
        leftover = ctx->leftover = lastline + 1 - buffer;
        leftoverlen = ctx->leftoverlen = length - leftover;
        *input = buffer;
        return leftover - 1;
    } else {
        // If a newline wasn't found at all, process the whole buffer next time
```

```
396         *input = 0;
397         if(length == MAX_BUFFER){
398             // Unless the buffer is completely full, in which case discard it
399             ckb_warn("Too much input (1MB). Dropping.\n");
400             return 0;
401         }
402         leftoverlen = ctx->leftoverlen = length;
403         return 0;
404     }
405 }
```

Here is the caller graph for this function:



5.9.3.5 void readlines_ctx_free (readlines_ctx *ctx*)

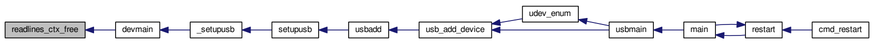
Definition at line 348 of file devnode.c.

References _readlines_ctx::buffer.

Referenced by devmain().

```
348                                           {
349     free(ctx->buffer);
350     free(ctx);
351 }
```

Here is the caller graph for this function:



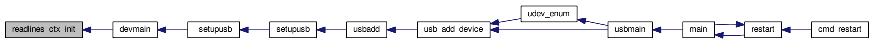
5.9.3.6 void readlines_ctx_init (readlines_ctx ∗ *ctx*)

Definition at line 341 of file devnode.c.

Referenced by devmain().

```
341                                             {
342     // Allocate buffers to store data
343     *ctx = calloc(1, sizeof(struct _readlines_ctx));
344     int buffersize = (*ctx)->buffersize = 4095;
345     (*ctx)->buffer = malloc(buffersize + 1);
346 }
```

Here is the caller graph for this function:



5.9.3.7 int rmdevpath (**usbdevice** ∗ *kb*)

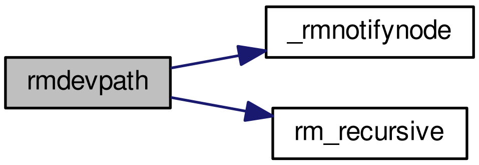
Definition at line 275 of file devnode.c.

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

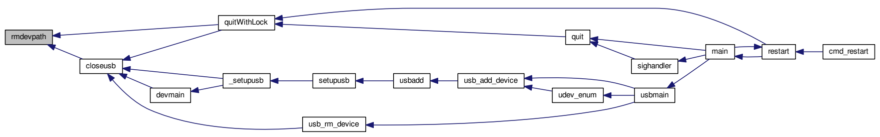
Referenced by closeusb(), and quitWithLock().

```
275                             {
276     euid_guard_start;
277     int index = INDEX_OF(kb, keyboard);
278     if(kb->infifo != 0){
279 #ifdef OS_LINUX
280         write(kb->infifo - 1, "\n", 1); // hack to prevent the FIFO thread from perma-blocking
281 #endif
282         close(kb->infifo - 1);
283         kb->infifo = 0;
284     }
285     for(int i = 0; i < OUTFIFO_MAX; i++)
286         _rmnotifynode(kb, i);
287     char path[strlen(devpath) + 2];
288     snprintf(path, sizeof(path), "%s%d", devpath, index);
289     if(rm_recursive(path) != 0 && errno != ENOENT){
290         ckb_warn("Unable to delete %s: %s\n", path, strerror(errno));
291         euid_guard_stop;
292         return -1;
293     }
294     ckb_info("Removed device path %s\n", path);
295     euid_guard_stop;
296     return 0;
297 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.9.3.8 int rmnotifynode (**usbdevice** ∗ *kb,* int *notify*)

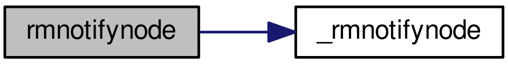
Definition at line 129 of file devnode.c.

References _rmnotifynode(), euid_guard_start, and euid_guard_stop.

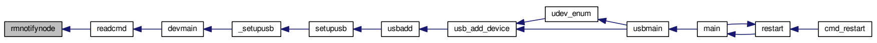
Referenced by readcmd().

```
                                            {
    euid_guard_start;
    int res = _rmnotifynode(kb, notify);
    euid_guard_stop;
    return res;
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.9.3.9 void updateconnected ( )

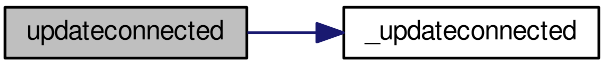
Definition at line 81 of file devnode.c.

References _updateconnected(), euid_guard_start, and euid_guard_stop.

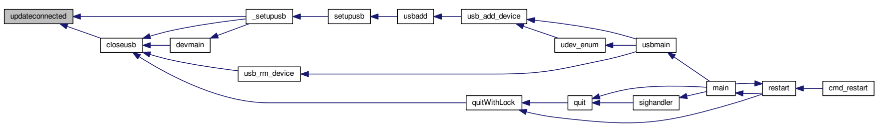
Referenced by _setupusb(), and closeusb().

```
                      {
    euid_guard_start;
    _updateconnected();
    euid_guard_stop;
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.9.4 Variable Documentation

5.9.4.1 const char∗ const devpath

Definition at line 8 of file devnode.h.

5.9.4.2 long gid

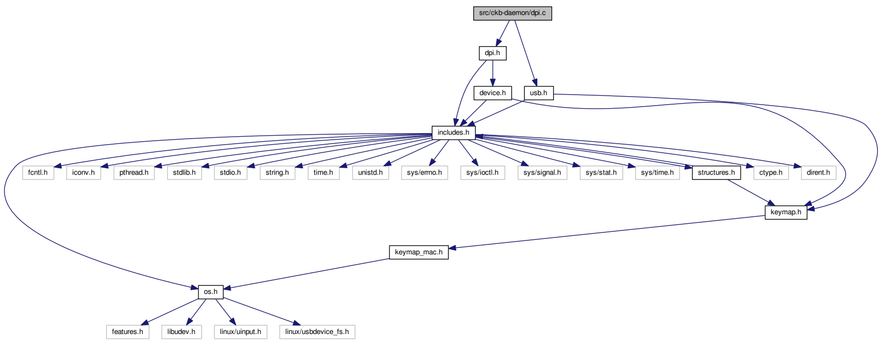
Definition at line 16 of file devnode.c.

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

5.10 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)

5.10.1 Function Documentation

5.10.1.1 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.

```
                                                                                        {
    int disable = 0;
    ushort x, y;
    // Try to scan X,Y values
    if(sscanf(values, "%hu,%hu", &x, &y) != 2){
        // If that doesn't work, scan single number
        if(sscanf(values, "%hu", &x) == 1)
            y = x;
        else if(!strncmp(values, "off", 3))
            // If the right side says "off", disable the level(s)
            disable = 1;
        else
            // Otherwise, quit
            return;
    }
    if((x == 0 || y == 0) && !disable)
        return;
    // Scan the left side for stage numbers (comma-separated)
    int left = strlen(stages);
    int position = 0, field = 0;
    char stagename[3];
    while(position < left && sscanf(stages + position, "%2[^,]%n", stagename, &field) == 1){
        uchar stagenum;
        if(sscanf(stagename, "%hhu", &stagenum) && stagenum < DPI_COUNT){
            // Set DPI for this stage
            if(disable){
                mode->dpi.enabled &= ~(1 << stagenum);
                mode->dpi.x[stagenum] = 0;
                mode->dpi.y[stagenum] = 0;
            } else {
                mode->dpi.enabled |= 1 << stagenum;
                mode->dpi.x[stagenum] = x;
                mode->dpi.y[stagenum] = y;
            }
        }
        if(stages[position += field] == ',')
            position++;
    }
}
```

5.10.1.2 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.

```
                                                                                                 {
    uchar stagenum;
    if(sscanf(stage, "%hhu", &stagenum) != 1)
        return;
    if(stagenum > DPI_COUNT)
        return;
    mode->dpi.current = stagenum;
}
```

5.10.1.3 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.

```
                                                                                                {
    uchar heightnum;
    if(sscanf(height, "%hhu", &heightnum) != 1)
        return;
    if(heightnum > LIFT_MAX || heightnum < LIFT_MIN)
        return;
    mode->dpi.lift = heightnum;
}
```

5.10.1.4 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.

```
                                                                                    {
    if(!strcmp(enable, "on"))
        mode->dpi.snap = 1;
    if(!strcmp(enable, "off"))
        mode->dpi.snap = 0;
}
```

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

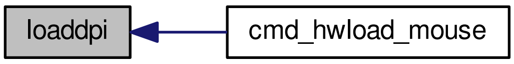
Definition at line 152 of file dpi.c.

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().

```
                                                         {
    // Ask for settings
    uchar data_pkt[4][MSG_SIZE] = {
        { 0x0e, 0x13, 0x05, 1, },
        { 0x0e, 0x13, 0x02, 1, },
        { 0x0e, 0x13, 0x03, 1, },
        { 0x0e, 0x13, 0x04, 1, }
    };
    uchar in_pkt[4][MSG_SIZE];
    for(int i = 0; i < 4; i++){
        if(!usbrecv(kb, data_pkt[i], in_pkt[i]))
            return -2;
        if(memcmp(in_pkt[i], data_pkt[i], 4)){
            ckb_err("Bad input header\n");
            return -3;
        }
    }
    // Copy data from device
    dpi->enabled = in_pkt[0][4];
    dpi->enabled &= (1 << DPI_COUNT) - 1;
    dpi->current = in_pkt[1][4];
    if(dpi->current >= DPI_COUNT)
        dpi->current = 0;
    dpi->lift = in_pkt[2][4];
    if(dpi->lift < LIFT_MIN || dpi->lift > LIFT_MAX)
        dpi->lift = LIFT_MIN;
    dpi->snap = !!in_pkt[3][4];

    // Get X/Y DPIs
    for(int i = 0; i < DPI_COUNT; i++){
        uchar data_pkt[MSG_SIZE] = { 0x0e, 0x13, 0xd0, 1 };
        uchar in_pkt[MSG_SIZE];
        data_pkt[2] |= i;
        if(!usbrecv(kb, data_pkt, in_pkt))
            return -2;
        if(memcmp(in_pkt, data_pkt, 4)){
            ckb_err("Bad input header\n");
            return -3;
        }
        // Copy to profile
        dpi->x[i] = *(ushort*)(in_pkt + 5);
        dpi->y[i] = *(ushort*)(in_pkt + 7);
        light->r[LED_MOUSE + N_MOUSE_ZONES + i] = in_pkt[9];
        light->g[LED_MOUSE + N_MOUSE_ZONES + i] = in_pkt[10];
        light->b[LED_MOUSE + N_MOUSE_ZONES + i] = in_pkt[11];
    }
    // Finished. Set SW DPI light to the current hardware level
    light->r[LED_MOUSE + 2] = light->r[LED_MOUSE +
  N_MOUSE_ZONES + dpi->current];
    light->g[LED_MOUSE + 2] = light->g[LED_MOUSE +
  N_MOUSE_ZONES + dpi->current];
    light->b[LED_MOUSE + 2] = light->b[LED_MOUSE +
  N_MOUSE_ZONES + dpi->current];
    return 0;
}
```

Here is the caller graph for this function:



5.10.1.6 char∗ printdpi (const dpiset ∗ *dpi*, const usbdevice ∗ *kb*)

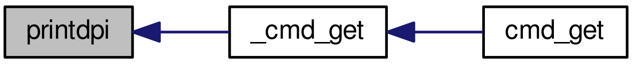
Definition at line 69 of file dpi.c.

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

Referenced by _cmd_get().

```
                                                    {
    // Print all DPI settings
    const int BUFFER_LEN = 100;
    char* buffer = malloc(BUFFER_LEN);
    int length = 0;
    for(int i = 0; i < DPI_COUNT; i++){
        // Print the stage number
        int newlen = 0;
        snprintf(buffer + length, BUFFER_LEN - length, length == 0 ? "%d%n" : " %d%n", i, &newlen);
        length += newlen;
        // Print the DPI settings
        if(!(dpi->enabled & (1 << i)))
            snprintf(buffer + length, BUFFER_LEN - length, ":off%n", &newlen);
        else
            snprintf(buffer + length, BUFFER_LEN - length, ":%u,%u%n", dpi->x[i], dpi->
   y[i], &newlen);
        length += newlen;
    }
    return buffer;
}
```

Here is the caller graph for this function:



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

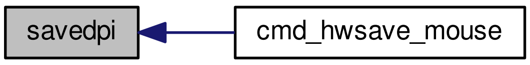
Definition at line 124 of file dpi.c.

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().

```
                                                        {
    // Send X/Y DPIs
    for(int i = 0; i < DPI_COUNT; i++){
        uchar data_pkt[MSG_SIZE] = { 0x07, 0x13, 0xd0, 1 };
        data_pkt[2] |= i;
        *(ushort*)(data_pkt + 5) = dpi->x[i];
        *(ushort*)(data_pkt + 7) = dpi->y[i];
        // Save the RGB value for this setting too
        data_pkt[9] = light->r[LED_MOUSE + N_MOUSE_ZONES + i];
        data_pkt[10] = light->g[LED_MOUSE + N_MOUSE_ZONES + i];
        data_pkt[11] = light->b[LED_MOUSE + N_MOUSE_ZONES + i];
        if(!usbsend(kb, data_pkt, 1))
            return -1;
    }

    // Send settings
    uchar data_pkt[4][MSG_SIZE] = {
        { 0x07, 0x13, 0x05, 1, dpi->enabled },
        { 0x07, 0x13, 0x02, 1, dpi->current },
        { 0x07, 0x13, 0x03, 1, dpi->lift },
        { 0x07, 0x13, 0x04, 1, dpi->snap, 0x05 }
    };
    if(!usbsend(kb, data_pkt[0], 4))
        return -2;
    // Finished
    return 0;
}
```

Here is the caller graph for this function:



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

Definition at line 89 of file dpi.c.

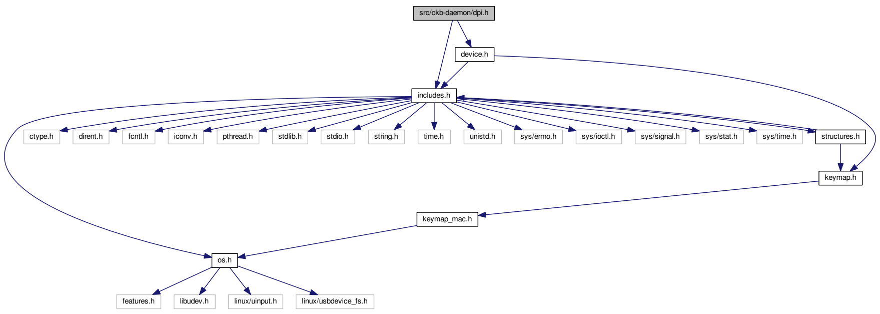
References usbdevice::active, dpiset::current, usbprofile::currentmode, usbmode::dpi, DPI_COUNT, dpiset::enabled, dpiset::forceupdate, usbprofile::lastdpi, dpiset::lift, MSG_SIZE, usbdevice::profile, dpiset::snap, usbsend, dpiset::x, and dpiset::y.

```
                                      {
    if(!kb->active)
        return 0;
    dpiset* lastdpi = &kb->profile->lastdpi;
    dpiset* newdpi = &kb->profile->currentmode->dpi;
    // Don't do anything if the settings haven't changed
    if(!force && !lastdpi->forceupdate && !newdpi->forceupdate
            && !memcmp(lastdpi, newdpi, sizeof(dpiset)))
        return 0;
    lastdpi->forceupdate = newdpi->forceupdate = 0;

    // Send X/Y DPIs
    for(int i = 0; i < DPI_COUNT; i++){
        uchar data_pkt[MSG_SIZE] = { 0x07, 0x13, 0xd0, 0 };
        data_pkt[2] |= i;
        *(ushort*)(data_pkt + 5) = newdpi->x[i];
        *(ushort*)(data_pkt + 7) = newdpi->y[i];
        if(!usbsend(kb, data_pkt, 1))
            return -1;
    }

    // Send settings
    uchar data_pkt[4][MSG_SIZE] = {
        { 0x07, 0x13, 0x05, 0, newdpi->enabled },
        { 0x07, 0x13, 0x02, 0, newdpi->current },
        { 0x07, 0x13, 0x03, 0, newdpi->lift },
        { 0x07, 0x13, 0x04, 0, newdpi->snap, 0x05 }
    };
    if(!usbsend(kb, data_pkt[0], 4))
        return -2;
    // Finished
    memcpy(lastdpi, newdpi, sizeof(dpiset));
    return 0;
}
```

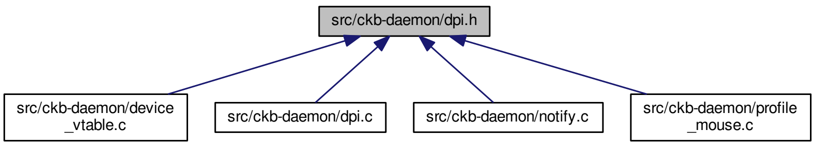
5.11 src/ckb-daemon/dpi.h File Reference

```
#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)

5.11.1 Function Documentation

5.11.1.1 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.

```
                                                                                                    {
     int disable = 0;
     ushort x, y;
     // Try to scan X,Y values
     if(sscanf(values, "%hu,%hu", &x, &y) != 2){
         // If that doesn't work, scan single number
         if(sscanf(values, "%hu", &x) == 1)
             y = x;
         else if(!strncmp(values, "off", 3))
             // If the right side says "off", disable the level(s)
             disable = 1;
         else
             // Otherwise, quit
             return;
     }
     if((x == 0 || y == 0) && !disable)
         return;
     // Scan the left side for stage numbers (comma-separated)
     int left = strlen(stages);
     int position = 0, field = 0;
     char stagename[3];
     while(position < left && sscanf(stages + position, "%2[^,]%n", stagename, &field) == 1){
         uchar stagenum;
         if(sscanf(stagename, "%hhu", &stagenum) && stagenum < DPI_COUNT){
             // Set DPI for this stage
             if(disable){
                 mode->dpi.enabled &= ~(1 << stagenum);
                 mode->dpi.x[stagenum] = 0;
                 mode->dpi.y[stagenum] = 0;
             } else {
                 mode->dpi.enabled |= 1 << stagenum;
                 mode->dpi.x[stagenum] = x;
                 mode->dpi.y[stagenum] = y;
             }
         }
         if(stages[position += field] == ',')
             position++;
     }
}
```

5.11.1.2 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.

```
                                                                                               {
     uchar stagenum;
     if(sscanf(stage, "%hhu", &stagenum) != 1)
         return;
     if(stagenum > DPI_COUNT)
         return;
     mode->dpi.current = stagenum;
}
```

5.11.1.3 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.

```
                                                                                              {
     uchar heightnum;
     if(sscanf(height, "%hhu", &heightnum) != 1)
         return;
     if(heightnum > LIFT_MAX || heightnum < LIFT_MIN)
         return;
     mode->dpi.lift = heightnum;
}
```

5.11.1.4 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.

```
                                                                                        {
      if(!strcmp(enable, "on"))
          mode->dpi.snap = 1;
      if(!strcmp(enable, "off"))
          mode->dpi.snap = 0;
  }
```

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

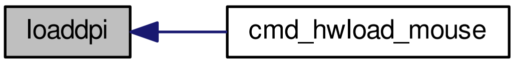
Definition at line 152 of file dpi.c.

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().

```
                                                         {
    // Ask for settings
    uchar data_pkt[4][MSG_SIZE] = {
        { 0x0e, 0x13, 0x05, 1, },
        { 0x0e, 0x13, 0x02, 1, },
        { 0x0e, 0x13, 0x03, 1, },
        { 0x0e, 0x13, 0x04, 1, }
    };
    uchar in_pkt[4][MSG_SIZE];
    for(int i = 0; i < 4; i++){
        if(!usbrecv(kb, data_pkt[i], in_pkt[i]))
            return -2;
        if(memcmp(in_pkt[i], data_pkt[i], 4)){
            ckb_err("Bad input header\n");
            return -3;
        }
    }
    // Copy data from device
    dpi->enabled = in_pkt[0][4];
    dpi->enabled &= (1 << DPI_COUNT) - 1;
    dpi->current = in_pkt[1][4];
    if(dpi->current >= DPI_COUNT)
        dpi->current = 0;
    dpi->lift = in_pkt[2][4];
    if(dpi->lift < LIFT_MIN || dpi->lift > LIFT_MAX)
        dpi->lift = LIFT_MIN;
    dpi->snap = !!in_pkt[3][4];

    // Get X/Y DPIs
    for(int i = 0; i < DPI_COUNT; i++){
        uchar data_pkt[MSG_SIZE] = { 0x0e, 0x13, 0xd0, 1 };
        uchar in_pkt[MSG_SIZE];
        data_pkt[2] |= i;
        if(!usbrecv(kb, data_pkt, in_pkt))
            return -2;
        if(memcmp(in_pkt, data_pkt, 4)){
            ckb_err("Bad input header\n");
            return -3;
        }
        // Copy to profile
        dpi->x[i] = *(ushort*)(in_pkt + 5);
        dpi->y[i] = *(ushort*)(in_pkt + 7);
        light->r[LED_MOUSE + N_MOUSE_ZONES + i] = in_pkt[9];
        light->g[LED_MOUSE + N_MOUSE_ZONES + i] = in_pkt[10];
        light->b[LED_MOUSE + N_MOUSE_ZONES + i] = in_pkt[11];
    }
    // Finished. Set SW DPI light to the current hardware level
    light->r[LED_MOUSE + 2] = light->r[LED_MOUSE +
  N_MOUSE_ZONES + dpi->current];
    light->g[LED_MOUSE + 2] = light->g[LED_MOUSE +
  N_MOUSE_ZONES + dpi->current];
    light->b[LED_MOUSE + 2] = light->b[LED_MOUSE +
  N_MOUSE_ZONES + dpi->current];
    return 0;
}
```

Here is the caller graph for this function:



5.11.1.6 char∗ printdpi (const dpiset ∗ *dpi,* const usbdevice ∗ *kb*)

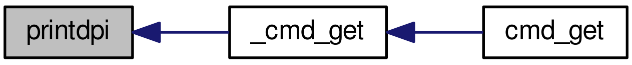
Definition at line 69 of file dpi.c.

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

Referenced by _cmd_get().

```
                                                    {
    // Print all DPI settings
    const int BUFFER_LEN = 100;
    char* buffer = malloc(BUFFER_LEN);
    int length = 0;
    for(int i = 0; i < DPI_COUNT; i++){
        // Print the stage number
        int newlen = 0;
        snprintf(buffer + length, BUFFER_LEN - length, length == 0 ? "%d%n" : " %d%n", i, &newlen);
        length += newlen;
        // Print the DPI settings
        if(!(dpi->enabled & (1 << i)))
            snprintf(buffer + length, BUFFER_LEN - length, ":off%n", &newlen);
        else
            snprintf(buffer + length, BUFFER_LEN - length, ":%u,%u%n", dpi->x[i], dpi->
      y[i], &newlen);
        length += newlen;
    }
    return buffer;
}
```

Here is the caller graph for this function:



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

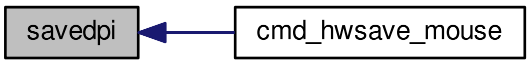
Definition at line 124 of file dpi.c.

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().

```
                                                        {
    // Send X/Y DPIs
    for(int i = 0; i < DPI_COUNT; i++){
        uchar data_pkt[MSG_SIZE] = { 0x07, 0x13, 0xd0, 1 };
        data_pkt[2] |= i;
        *(ushort*)(data_pkt + 5) = dpi->x[i];
        *(ushort*)(data_pkt + 7) = dpi->y[i];
        // Save the RGB value for this setting too
        data_pkt[9] = light->r[LED_MOUSE + N_MOUSE_ZONES + i];
        data_pkt[10] = light->g[LED_MOUSE + N_MOUSE_ZONES + i];
        data_pkt[11] = light->b[LED_MOUSE + N_MOUSE_ZONES + i];
        if(!usbsend(kb, data_pkt, 1))
            return -1;
    }

    // Send settings
    uchar data_pkt[4][MSG_SIZE] = {
        { 0x07, 0x13, 0x05, 1, dpi->enabled },
        { 0x07, 0x13, 0x02, 1, dpi->current },
        { 0x07, 0x13, 0x03, 1, dpi->lift },
        { 0x07, 0x13, 0x04, 1, dpi->snap, 0x05 }
    };
    if(!usbsend(kb, data_pkt[0], 4))
        return -2;
    // Finished
    return 0;
}
```

Here is the caller graph for this function:



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

Definition at line 89 of file dpi.c.

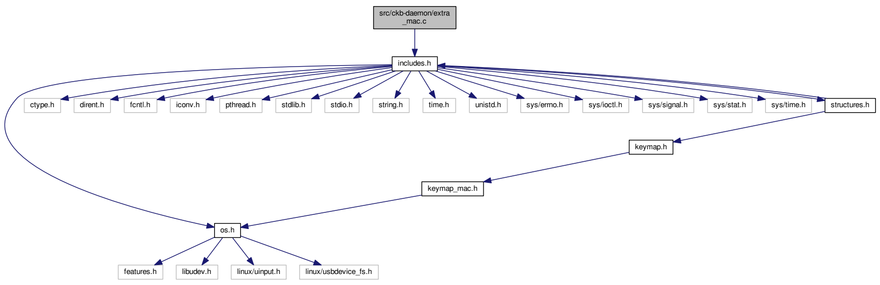
References usbdevice::active, dpiset::current, usbprofile::currentmode, usbmode::dpi, DPI_COUNT, dpiset::enabled, dpiset::forceupdate, usbprofile::lastdpi, dpiset::lift, MSG_SIZE, usbdevice::profile, dpiset::snap, usbsend, dpiset::x, and dpiset::y.

```
                                        {
    if(!kb->active)
        return 0;
    dpiset* lastdpi = &kb->profile->lastdpi;
    dpiset* newdpi = &kb->profile->currentmode->dpi;
    // Don't do anything if the settings haven't changed
    if(!force && !lastdpi->forceupdate && !newdpi->forceupdate
            && !memcmp(lastdpi, newdpi, sizeof(dpiset)))
        return 0;
    lastdpi->forceupdate = newdpi->forceupdate = 0;

    // Send X/Y DPIs
    for(int i = 0; i < DPI_COUNT; i++){
        uchar data_pkt[MSG_SIZE] = { 0x07, 0x13, 0xd0, 0 };
        data_pkt[2] |= i;
        *(ushort*)(data_pkt + 5) = newdpi->x[i];
        *(ushort*)(data_pkt + 7) = newdpi->y[i];
        if(!usbsend(kb, data_pkt, 1))
            return -1;
    }

    // Send settings
    uchar data_pkt[4][MSG_SIZE] = {
        { 0x07, 0x13, 0x05, 0, newdpi->enabled },
        { 0x07, 0x13, 0x02, 0, newdpi->current },
        { 0x07, 0x13, 0x03, 0, newdpi->lift },
        { 0x07, 0x13, 0x04, 0, newdpi->snap, 0x05 }
    };
    if(!usbsend(kb, data_pkt[0], 4))
        return -2;
    // Finished
    memcpy(lastdpi, newdpi, sizeof(dpiset));
    return 0;
}
```

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

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

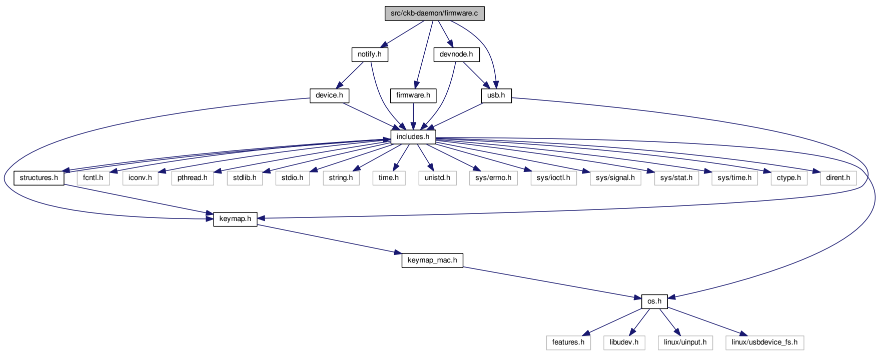
Include dependency graph for extra_mac.c:



5.13 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)

5.13.1 Macro Definition Documentation

5.13.1.1 #define FW_MAXSIZE (255 ∗ 256)

Definition at line 51 of file firmware.c.

Referenced by fwupdate().

5.13.1.2 #define FW_NOFILE -1

Definition at line 7 of file firmware.c.

Referenced by cmd_fwupdate(), and fwupdate().

5.13.1.3 #define FW_OK 0

Definition at line 6 of file firmware.c.

Referenced by cmd_fwupdate(), and fwupdate().

5.13.1.4 #define FW_USBFAIL -3

Definition at line 9 of file firmware.c.

Referenced by cmd_fwupdate(), and fwupdate().

5.13.1.5 #define FW_WRONGDEV -2

Definition at line 8 of file firmware.c.

Referenced by cmd_fwupdate(), and fwupdate().

5.13.2 Function Documentation

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

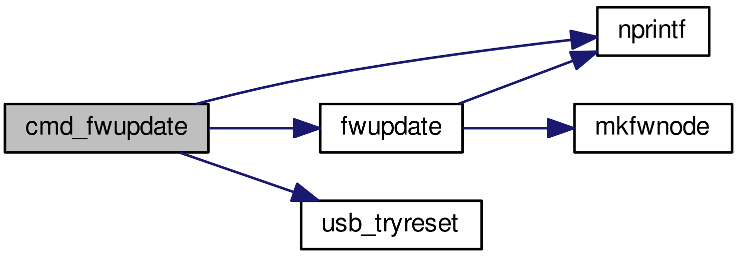
Definition at line 154 of file firmware.c.

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

```
                                                                                        {
    if(!HAS_FEATURES(kb, FEAT_FWUPDATE))
        return 0;
    // Update the firmware
    int ret = fwupdate(kb, path, nnumber);
    while(ret == FW_USBFAIL){
        // Try to reset the device if it fails
        if(usb_tryreset(kb))
            break;
        ret = fwupdate(kb, path, nnumber);
    }
    switch(ret){
    case FW_OK:
        nprintf(kb, nnumber, 0, "fwupdate %s ok\n", path);
```

```
            break;
        case FW_NOFILE:
        case FW_WRONGDEV:
            nprintf(kb, nnumber, 0, "fwupdate %s invalid\n", path);
            break;
        case FW_USBFAIL:
            nprintf(kb, nnumber, 0, "fwupdate %s fail\n", path);
            return -1;
        }
        return 0;
}
```

Here is the call graph for this function:



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

Definition at line 55 of file firmware.c.

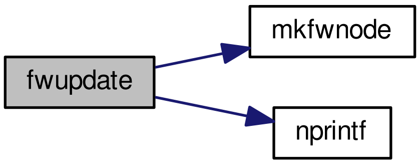
References ckb_err, ckb_info, FW_MAXSIZE, FW_NOFILE, FW_OK, FW_USBFAIL, FW_WRONGDEV, usbdevice::fwversion, mkfwnode(), MSG_SIZE, nprintf(), usbdevice::product, usbdevice::usbdelay, usbsend, and usbdevice::vendor.

Referenced by cmd_fwupdate().

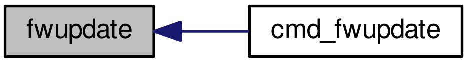
```
                                                          {
    // Read the firmware from the given path
    char* fwdata = calloc(1, FW_MAXSIZE + 256);
    int fd = open(path, O_RDONLY);
    if(fd == -1){
        ckb_err("Failed to open firmware file %s: %s\n", path, strerror(errno));
        return FW_NOFILE;
    }
    ssize_t length = read(fd, fwdata, FW_MAXSIZE + 1);
    if(length <= 0x108 || length > FW_MAXSIZE){
        ckb_err("Failed to read firmware file %s: %s\n", path, length <= 0 ? strerror(errno) : "
      Wrong size");
        close(fd);
        return FW_NOFILE;
    }
    close(fd);

    short vendor, product, version;
    // Copy the vendor ID, product ID, and version from the firmware file
    memcpy(&vendor, fwdata + 0x102, 2);
    memcpy(&product, fwdata + 0x104, 2);
    memcpy(&version, fwdata + 0x106, 2);
    // Check against the actual device
    if(vendor != kb->vendor || product != kb->product){
        ckb_err("Firmware file %s doesn't match device (V: %04x P: %04x)\n", path, vendor, product);
        return FW_WRONGDEV;
    }
    ckb_info("Loading firmware version %04x from %s\n", version, path);
    nprintf(kb, nnumber, 0, "fwupdate %s 0/%d\n", path, (int)length);
    // Force the device to 10ms delay (we need to deliver packets very slowly to make sure it doesn't get
    overwhelmed)
```

```
   kb->usbdelay = 10;
   // Send the firmware messages (256 bytes at a time)
   uchar data_pkt[7][MSG_SIZE] = {
       { 0x07, 0x0c, 0xf0, 0x01, 0 },
       { 0x07, 0x0d, 0xf0, 0 },
       { 0x7f, 0x01, 0x3c, 0 },
       { 0x7f, 0x02, 0x3c, 0 },
       { 0x7f, 0x03, 0x3c, 0 },
       { 0x7f, 0x04, 0x3c, 0 },
       { 0x7f, 0x05, 0x10, 0 }
   };
   int output = 0, last = 0;
   int index = 0;
   while(output < length){
       int npackets = 1;
       // Packet 1: data position
        data_pkt[1][6] = index++;
        while(output < length){
            npackets++;
            if(npackets != 6){
                // Packets 2-5: 60 bytes of data
                memcpy(data_pkt[npackets] + 4, fwdata + output, 60);
                last = output;
                output += 60;
            } else {
                // Packet 6: 16 bytes
                memcpy(data_pkt[npackets] + 4, fwdata + output, 16);
                last = output;
                output += 16;
                break;
            }
        }
        if(index == 1){
            if(!usbsend(kb, data_pkt[0], 1)){
                ckb_err("Firmware update failed\n");
                return FW_USBFAIL;
            }
            // The above packet can take a lot longer to process, so wait for a while
            sleep(3);
            if(!usbsend(kb, data_pkt[2], npackets - 1)){
                ckb_err("Firmware update failed\n");
                return FW_USBFAIL;
            }
        } else {
            // If the output ends here, set the length byte appropriately
            if(output >= length)
                data_pkt[npackets][2] = length - last;
            if(!usbsend(kb, data_pkt[1], npackets)){
                ckb_err("Firmware update failed\n");
                return FW_USBFAIL;
            }
        }
        nprintf(kb, nnumber, 0, "fwupdate %s %d/%d\n", path, output, (int)length);
    }
    // Send the final pair of messages
    uchar data_pkt2[2][MSG_SIZE] = {
        { 0x07, 0x0d, 0xf0, 0x00, 0x00, index },
        { 0x07, 0x02, 0xf0, 0 }
    };
    if(!usbsend(kb, data_pkt2[0], 2)){
        ckb_err("Firmware update failed\n");
        return FW_USBFAIL;
    }
    // Updated successfully
    kb->fwversion = version;
    mkfwnode(kb);
    ckb_info("Firmware update complete\n");
    return FW_OK;
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.13.2.3 int getfwversion (usbdevice * *kb*)

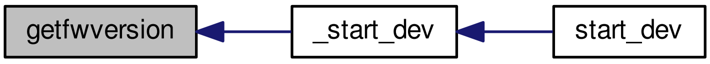
Definition at line 11 of file firmware.c.

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().

```
                                {
    // Ask board for firmware info
    uchar data_pkt[MSG_SIZE] = { 0x0e, 0x01, 0 };
    uchar in_pkt[MSG_SIZE];
    if(!usbrecv(kb, data_pkt, in_pkt))
        return -1;
    if(in_pkt[0] != 0x0e || in_pkt[1] != 0x01){
        ckb_err("Bad input header\n");
        return -1;
    }
    short vendor, product, version, bootloader;
    // Copy the vendor ID, product ID, version, and poll rate from the firmware data
    memcpy(&version, in_pkt + 8, 2);
    memcpy(&bootloader, in_pkt + 10, 2);
    memcpy(&vendor, in_pkt + 12, 2);
    memcpy(&product, in_pkt + 14, 2);
    char poll = in_pkt[16];
    if(poll <= 0){
        poll = -1;
        kb->features &= ~FEAT_POLLRATE;
    }
    // Print a warning if the message didn't match the expected data
    if(vendor != kb->vendor)
        ckb_warn("Got vendor ID %04x (expected %04x)\n", vendor, kb->
   vendor);
    if(product != kb->product)
        ckb_warn("Got product ID %04x (expected %04x)\n", product, kb->
   product);
    // Set firmware version and poll rate
    if(version == 0 || bootloader == 0){
        // Needs firmware update
        kb->fwversion = 0;
        kb->pollrate = -1;
    } else {
        if(version != kb->fwversion && kb->fwversion != 0)
            ckb_warn("Got firmware version %04x (expected %04x)\n", version, kb->
    fwversion);
        kb->fwversion = version;
        kb->pollrate = poll;
    }
    return 0;
}
```

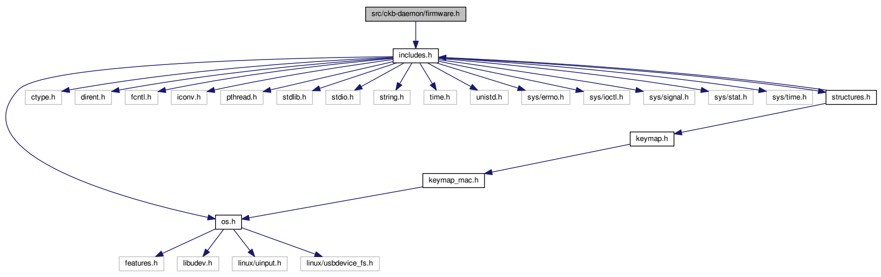
Here is the caller graph for this function:



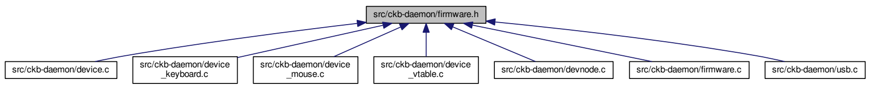
5.14 src/ckb-daemon/firmware.h File Reference

`#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)

5.14.1 Function Documentation

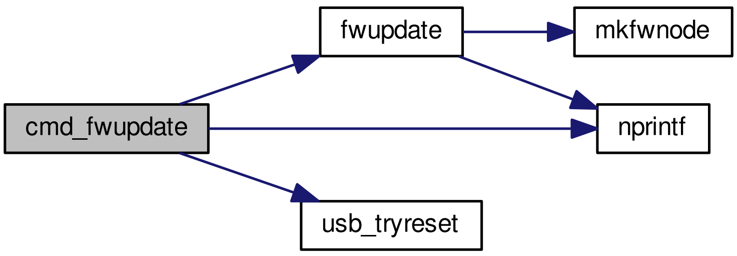
5.14.1.1 int cmd_fwupdate (usbdevice ∗ *kb,* usbmode ∗ *dummy1,* int *nnumber,* int *dummy2,* const char ∗ *path*)

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().

```
                                                                                             {
    if(!HAS_FEATURES(kb, FEAT_FWUPDATE))
        return 0;
    // Update the firmware
    int ret = fwupdate(kb, path, nnumber);
    while(ret == FW_USBFAIL){
        // Try to reset the device if it fails
        if(usb_tryreset(kb))
            break;
        ret = fwupdate(kb, path, nnumber);
    }
    switch(ret){
    case FW_OK:
        nprintf(kb, nnumber, 0, "fwupdate %s ok\n", path);
        break;
    case FW_NOFILE:
    case FW_WRONGDEV:
        nprintf(kb, nnumber, 0, "fwupdate %s invalid\n", path);
        break;
    case FW_USBFAIL:
        nprintf(kb, nnumber, 0, "fwupdate %s fail\n", path);
        return -1;
    }
    return 0;
}
```

Here is the call graph for this function:



5.14.1.2 int getfwversion (usbdevice ∗ *kb*)

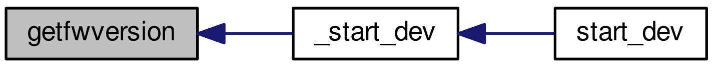
Definition at line 11 of file firmware.c.

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().

```
                              {
    // Ask board for firmware info
    uchar data_pkt[MSG_SIZE] = { 0x0e, 0x01, 0 };
    uchar in_pkt[MSG_SIZE];
    if(!usbrecv(kb, data_pkt, in_pkt))
        return -1;
    if(in_pkt[0] != 0x0e || in_pkt[1] != 0x01){
        ckb_err("Bad input header\n");
        return -1;
    }
    short vendor, product, version, bootloader;
    // Copy the vendor ID, product ID, version, and poll rate from the firmware data
    memcpy(&version, in_pkt + 8, 2);
    memcpy(&bootloader, in_pkt + 10, 2);
    memcpy(&vendor, in_pkt + 12, 2);
    memcpy(&product, in_pkt + 14, 2);
    char poll = in_pkt[16];
    if(poll <= 0){
        poll = -1;
        kb->features &= ~FEAT_POLLRATE;
    }
    // Print a warning if the message didn't match the expected data
    if(vendor != kb->vendor)
        ckb_warn("Got vendor ID %04x (expected %04x)\n", vendor, kb->vendor);
    if(product != kb->product)
        ckb_warn("Got product ID %04x (expected %04x)\n", product, kb->product);
    // Set firmware version and poll rate
    if(version == 0 || bootloader == 0){
        // Needs firmware update
        kb->fwversion = 0;
        kb->pollrate = -1;
    } else {
        if(version != kb->fwversion && kb->fwversion != 0)
            ckb_warn("Got firmware version %04x (expected %04x)\n", version, kb->fwversion);
        kb->fwversion = version;
        kb->pollrate = poll;
    }
    return 0;
}
```

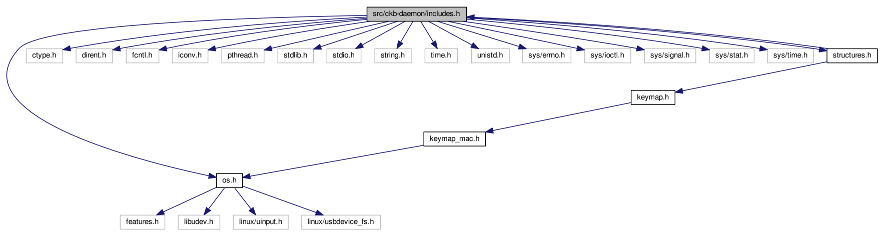
Here is the caller graph for this function:



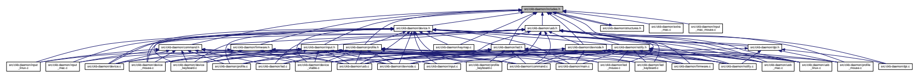
5.15 src/ckb-daemon/includes.h File Reference

```
#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)

5.15.1 Macro Definition Documentation

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

Definition at line 40 of file includes.h.

5.15.1.2 #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().

5.15.1.3 #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().

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

Definition at line 47 of file includes.h.

5.15.1.5 #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().

5.15.1.6 #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.

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

Definition at line 44 of file includes.h.

Referenced by main().

5.15.1.8 #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().

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

Definition at line 54 of file includes.h.

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

Definition at line 53 of file includes.h.

Referenced by main().

5.15.1.11 #define ckb_s_err stdout

Definition at line 36 of file includes.h.

5.15.1.12 #define ckb_s_out stdout

Definition at line 35 of file includes.h.

5.15.1.13 #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().

5.15.1.14 #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().

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

Definition at line 50 of file includes.h.

Referenced by main().

5.15.1.16 #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().

5.15.1.17 #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.

5.15.1.18 #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.

5.15.1.19 #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.

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

Definition at line 63 of file includes.h.

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

Definition at line 62 of file includes.h.

5.15.2 Typedef Documentation

5.15.2.1 typedef unsigned char uchar

Definition at line 24 of file includes.h.

5.15.2.2 typedef unsigned short ushort

Definition at line 25 of file includes.h.

5.15.3 Function Documentation

5.15.3.1 void timespec_add (struct timespec ∗ *timespec,* long *nanoseconds*)

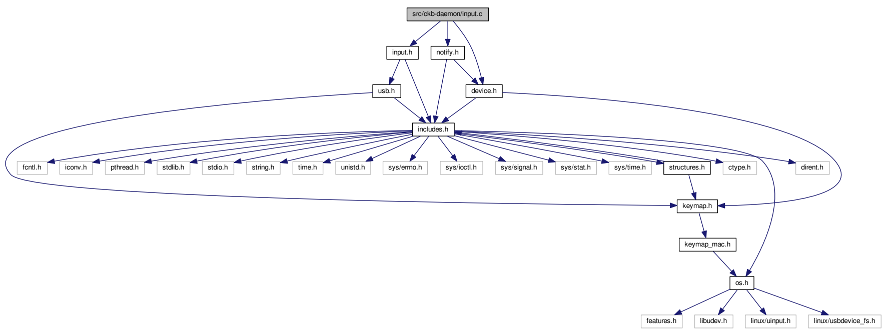
Definition at line 19 of file main.c.

```
                                                                {
    nanoseconds += timespec->tv_nsec;
    timespec->tv_sec += nanoseconds / 1000000000;
    timespec->tv_nsec = nanoseconds % 1000000000;
}
```

5.16 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)

5.16.1 Macro Definition Documentation

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

Referenced by inputupdate_keys().

5.16.2 Function Documentation

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

Definition at line 226 of file input.c.

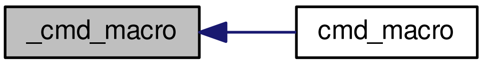
References keymacro::actioncount, keymacro::actions, usbmode::bind, keymacro::combo, macroaction::down, keymap, MACRO_MAX, binding::macrocap, binding::macrocount, binding::macros, N_KEYBYTES_INPUT, N_KEYS_INPUT, macroaction::scan, key::scan, and SET_KEYBIT.

Referenced by cmd_macro().

```
                                                                             {
  binding* bind = &mode->bind;
  if(!keys && !assignment){
      // Null strings = "macro clear" -> erase the whole thing
      for(int i = 0; i < bind->macrocount; i++)
          free(bind->macros[i].actions);
      bind->macrocount = 0;
      return;
  }
  if(bind->macrocount >= MACRO_MAX)
      return;
  // Create a key macro
  keymacro macro;
  memset(&macro, 0, sizeof(macro));
  // Scan the left side for key names, separated by +
  int empty = 1;
  int left = strlen(keys), right = strlen(assignment);
  int position = 0, field = 0;
  char keyname[12];
  while(position < left && sscanf(keys + position, "%10[^+]%n", keyname, &field) == 1){
      int keycode;
      if((sscanf(keyname, "#%d", &keycode) && keycode >= 0 && keycode <
N_KEYS_INPUT)
                || (sscanf(keyname, "#x%x", &keycode) && keycode >= 0 && keycode <
N_KEYS_INPUT)){
          // Set a key numerically
          SET_KEYBIT(macro.combo, keycode);
          empty = 0;
      } else {
          // Find this key in the keymap
          for(unsigned i = 0; i < N_KEYS_INPUT; i++){
              if(keymap[i].name && !strcmp(keyname, keymap[i].name)){
                  macro.combo[i / 8] |= 1 << (i % 8);
                  empty = 0;
                  break;
              }
          }
      }
      if(keys[position += field] == '+')
          position++;
  }
  if(empty)
      return;
  // Count the number of actions (comma separated)
  int count = 1;
  for(const char* c = assignment; *c != 0; c++){
      if(*c == ',')
          count++;
  }
  // Allocate a buffer for them
  macro.actions = calloc(count, sizeof(macroaction));
  macro.actioncount = 0;
  // Scan the actions
  position = 0;
  field = 0;
  while(position < right && sscanf(assignment + position, "%11[^,]%n", keyname, &field) == 1){
      if(!strcmp(keyname, "clear"))
          break;
      int down = (keyname[0] == '+');
      if(down || keyname[0] == '-'){
          int keycode;
          if((sscanf(keyname + 1, "#%d", &keycode) && keycode >= 0 && keycode < N_KEYS_INPUT)
                    || (sscanf(keyname + 1, "#x%x", &keycode) && keycode >= 0 && keycode <
N_KEYS_INPUT)){
              // Set a key numerically
              macro.actions[macro.actioncount].scan =
keymap[keycode].scan;
              macro.actions[macro.actioncount].down = down;
              macro.actioncount++;
          } else {
              // Find this key in the keymap
              for(unsigned i = 0; i < N_KEYS_INPUT; i++){
                  if(keymap[i].name && !strcmp(keyname + 1, keymap[i].name)){
                      macro.actions[macro.actioncount].scan =
keymap[i].scan;
                      macro.actions[macro.actioncount].down = down;
                      macro.actioncount++;
                      break;
                  }
              }
          }
      }
      if(assignment[position += field] == ',')
          position++;
  }

  // See if there's already a macro with this trigger
```

```
    keymacro* macros = bind->macros;
    for(int i = 0; i < bind->macrocount; i++){
        if(!memcmp(macros[i].combo, macro.combo, N_KEYBYTES_INPUT)){
            free(macros[i].actions);
            // If the new macro has no actions, erase the existing one
            if(!macro.actioncount){
                for(int j = i + 1; j < bind->macrocount; j++)
                    memcpy(macros + j - 1, macros + j, sizeof(keymacro));
                bind->macrocount--;
            } else
                // If there are actions, replace the existing with the new
                memcpy(macros + i, &macro, sizeof(keymacro));
            return;
        }
    }

    // Add the macro to the device settings if not empty
    if(macro.actioncount < 1)
        return;
    memcpy(bind->macros + (bind->macrocount++), &macro, sizeof(
  keymacro));
    if(bind->macrocount >= bind->macrocap)
        bind->macros = realloc(bind->macros, (bind->macrocap += 16) * sizeof(
  keymacro));
}
```

Here is the caller graph for this function:



5.16.2.2 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.

```
                                                                                  {
    if(keyindex >= N_KEYS_INPUT)
        return;
    // Find the key to bind to
    int tocode = 0;
    if(sscanf(to, "#x%ux", &tocode) != 1 && sscanf(to, "#%u", &tocode) == 1 && tocode <
  N_KEYS_INPUT){
        pthread_mutex_lock(imutex(kb));
        mode->bind.base[keyindex] = tocode;
        pthread_mutex_unlock(imutex(kb));
        return;
    }
    // If not numeric, look it up
    for(int i = 0; i < N_KEYS_INPUT; i++){
        if(keymap[i].name && !strcmp(to, keymap[i].name)){
            pthread_mutex_lock(imutex(kb));
            mode->bind.base[keyindex] = keymap[i].scan;
            pthread_mutex_unlock(imutex(kb));
            return;
        }
    }
}
```

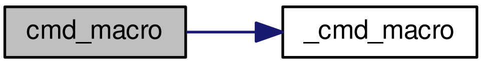
5.16.2.3 void cmd_macro (usbdevice ∗ *kb,* usbmode ∗ *mode,* const int *notifynumber,* const char ∗ *keys,* const char ∗ *assignment*)

Definition at line 332 of file input.c.

References _cmd_macro(), and imutex.

```
{
    pthread_mutex_lock(imutex(kb));
    _cmd_macro(mode, keys, assignment);
    pthread_mutex_unlock(imutex(kb));
}
```

Here is the call graph for this function:



5.16.2.4 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.

```
{
    if(keyindex >= N_KEYS_INPUT)
        return;
    pthread_mutex_lock(imutex(kb));
    mode->bind.base[keyindex] = keymap[keyindex].scan;
    pthread_mutex_unlock(imutex(kb));
}
```

5.16.2.5 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.

```
{
    if(keyindex >= N_KEYS_INPUT)
        return;
    pthread_mutex_lock(imutex(kb));
    mode->bind.base[keyindex] = KEY_UNBOUND;
    pthread_mutex_unlock(imutex(kb));
}
```

5.16.2.6 void freebind (**binding** ∗ *bind*)

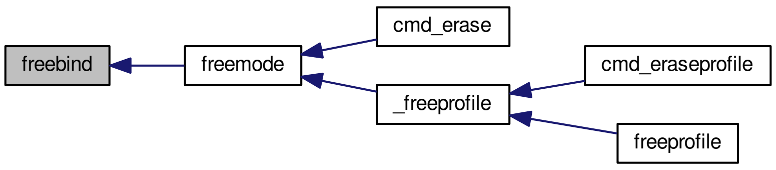
Definition at line 181 of file input.c.

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

Referenced by freemode().

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

Here is the caller graph for this function:



5.16.2.7 void initbind (**binding** ∗ *bind*)

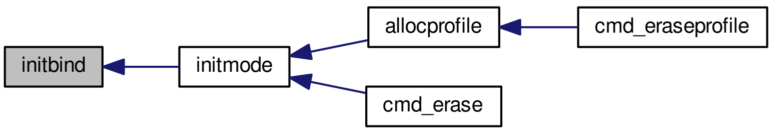
Definition at line 173 of file input.c.

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

Referenced by initmode().

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

Here is the caller graph for this function:



5.16.2.8 void inputupdate (**usbdevice** ∗ *kb*)

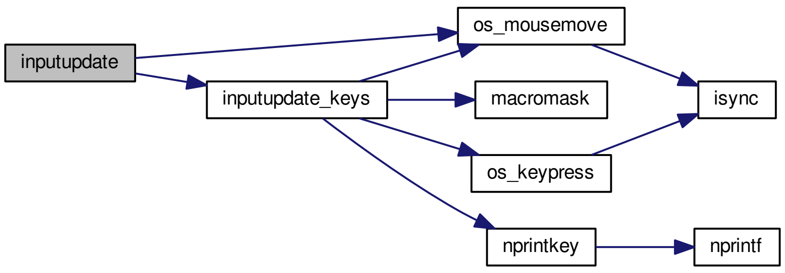
Definition at line 122 of file input.c.

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

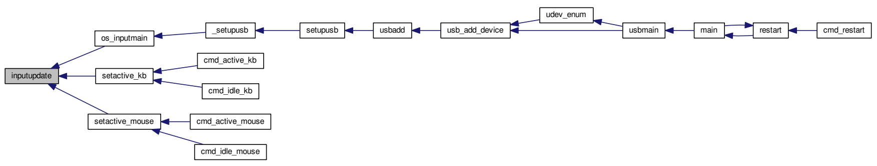
Referenced by os_inputmain(), setactive_kb(), and setactive_mouse().

```
                                {
#ifdef OS_LINUX
    if((!kb->uinput_kb || !kb->uinput_mouse)
#else
    if(!kb->event
#endif
            || !kb->profile)
        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:



5.16.2.9 static void inputupdate_keys (**usbdevice** ∗ *kb*) [static]

Definition at line 15 of file input.c.

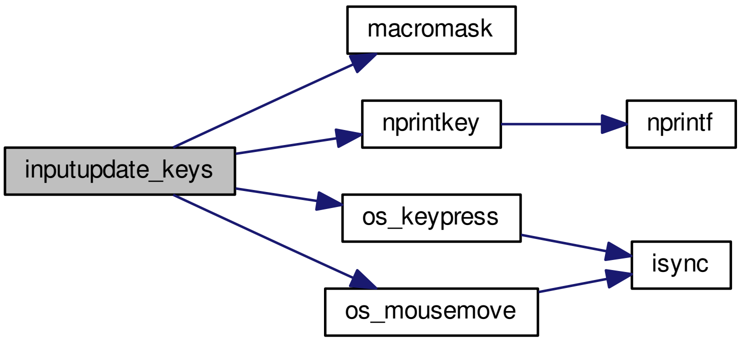
References keymacro::actioncount, keymacro::actions, usbdevice::active, binding::base, usbmode::bind, keymacro::combo, usbprofile::currentmode, usbdevice::delay, macroaction::down, usbdevice::input, IS_MOD, IS_WHEEL, keymap, usbinput::keys, binding::macrocount, macromask(), binding::macros, N_KEYBYTES_INPUT, 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().

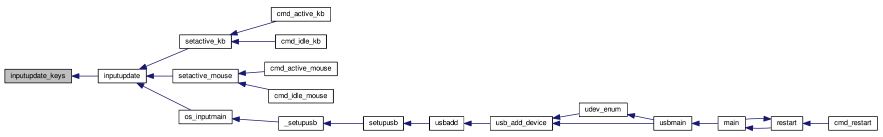
```
15                                         {
16     usbmode* mode = kb->profile->currentmode;
17     binding* bind = &mode->bind;
18     usbinput* input = &kb->input;
19     // Don't do anything if the state hasn't changed
20     if(!memcmp(input->prevkeys, input->keys, N_KEYBYTES_INPUT))
21         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++){
```

```
            keymacro* macro = &bind->macros[i];
            if(macromask(input->keys, macro->combo)){
                if(!macro->triggered){
                    macrotrigger = 1;
                    macro->triggered = 1;
                    // Send events for each keypress in the macro
                    for(int a = 0; a < macro->actioncount; a++){
                        macroaction* action = macro->actions + a;
                        if(action->rel_x != 0 || action->rel_y != 0)
                            os_mousemove(kb, action->rel_x, action->rel_y);
                        else {
                            os_keypress(kb, action->scan, action->down);
                            if (kb->delay) {
                                if (a > 200) usleep (100);
                                else if (a > 20) usleep(30);
                            }
                        }
                    }
                }
            } else {
                macro->triggered = 0;
            }
        }
    }
    // Make a list of keycodes to send. Rearrange them so that modifier keydowns always come first
    // and modifier keyups always come last. This ensures that shortcut keys will register properly
    // even if both keydown events happen at once.
    // N_KEYS + 4 is used because the volume wheel generates keydowns and keyups at the same time
    // (it's currently impossible to press all four at once, but safety first)
    int events[N_KEYS_INPUT + 4];
    int modcount = 0, keycount = 0, rmodcount = 0;
    for(int byte = 0; byte < N_KEYBYTES_INPUT; byte++){
        char oldb = input->prevkeys[byte], newb = input->keys[byte];
        if(oldb == newb)
            continue;
        for(int bit = 0; bit < 8; bit++){
            int keyindex = byte * 8 + bit;
            if(keyindex >= N_KEYS_INPUT)
                break;
            const key* map = keymap + keyindex;
            int scancode = (kb->active) ? bind->base[keyindex] : map->scan;
            char mask = 1 << bit;
            char old = oldb & mask, new = newb & mask;
            // If the key state changed, send it to the input device
            if(old != new){
                // Don't echo a key press if a macro was triggered or if there's no scancode associated
                if(!macrotrigger && !(scancode & SCAN_SILENT)){
                    if(IS_MOD(scancode)){
                        if(new){
                            // Modifier down: Add to the end of modifier keys
                            for(int i = keycount + rmodcount; i > 0; i--)
                                events[modcount + i] = events[modcount + i - 1];
                            // Add 1 to the scancode because A is zero on OSX
                            // Positive code = keydown, negative code = keyup
                            events[modcount++] = scancode + 1;
                        } else {
                            // Modifier up: Add to the end of everything
                            events[modcount + keycount + rmodcount++] = -(scancode + 1);
                        }
                    } else {
                        // Regular keypress: add to the end of regular keys
                        for(int i = rmodcount; i > 0; i--)
                            events[modcount + keycount + i] = events[modcount + keycount + i - 1];
                        events[modcount + keycount++] = new ? (scancode + 1) : -(scancode + 1);
                        // The volume wheel and the mouse wheel don't generate keyups, so create them automatically
#define IS_WHEEL(scan, kb)  (((scan) == KEY_VOLUMEUP || (scan) == KEY_VOLUMEDOWN || (scan) == BTN_WHEELUP || (scan) == BTN_WHEELDOWN) && !IS_K65(kb))
                        if(new && IS_WHEEL(map->scan, kb)){
                            for(int i = rmodcount; i > 0; i--)
                                events[modcount + keycount + i] = events[modcount + keycount + i - 1];
                            events[modcount + keycount++] = -(scancode + 1);
                            input->keys[byte] &= ~mask;
                        }
                    }
                }
                // Print notifications if desired
                if(kb->active){
                    for(int notify = 0; notify < OUTFIFO_MAX; notify++){
                        if(mode->notify[notify][byte] & mask){
                            nprintkey(kb, notify, keyindex, new);
                            // Wheels doesn't generate keyups
                            if(new && IS_WHEEL(map->scan, kb))
                                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:



5.16.2.10 int macromask (const uchar ∗ *key1,* const uchar ∗ *key2*)

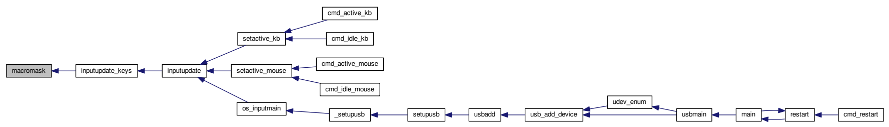
Definition at line 5 of file input.c.

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:



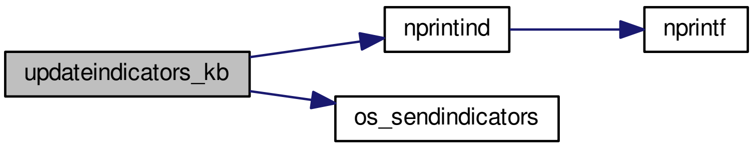
5.16.2.11 void updateindicators_kb (usbdevice ∗ *kb*, int *force*)

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.

```
                                                  {
    // Read current hardware indicator state (set externally)
    uchar old = kb->ileds, hw_old = kb->hw_ileds_old;
    uchar new = kb->hw_ileds, hw_new = new;
    // Update them if needed
    if(kb->active){
        usbmode* mode = kb->profile->currentmode;
        new = (new & ~mode->ioff) | mode->ion;
    }
    kb->ileds = new;
    kb->hw_ileds_old = hw_new;
    if(old != new || force){
        DELAY_SHORT(kb);
        os_sendindicators(kb);
    }
    // Print notifications if desired
    if(!kb->active)
        return;
    usbmode* mode = kb->profile->currentmode;
    uchar indicators[] = { I_NUM, I_CAPS, I_SCROLL };
    for(unsigned i = 0; i < sizeof(indicators) / sizeof(uchar); i++){
        uchar mask = indicators[i];
        if((hw_old & mask) == (hw_new & mask))
            continue;
        for(int notify = 0; notify < OUTFIFO_MAX; notify++){
            if(mode->inotify[notify] & mask)
                nprintind(kb, notify, mask, hw_new & mask);
        }
    }
}
```

Here is the call graph for this function:

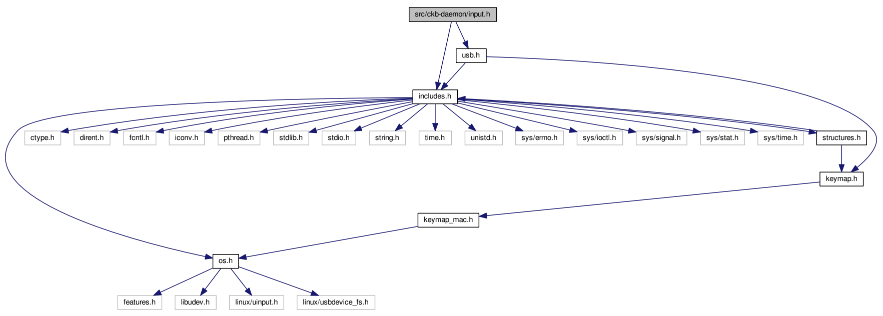


5.17 src/ckb-daemon/input.h File Reference

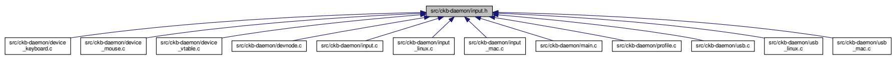
```
#include "includes.h"
```

`#include "usb.h"`

Include dependency graph for input.h:



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


src/ckb-daemon/input.h
src/ckb-daemon/device_keyboard.c
src/ckb-daemon/device_mouse.c
src/ckb-daemon/device_vtable.c
src/ckb-daemon/devnode.c
src/ckb-daemon/input.c
src/ckb-daemon/input_linux.c
src/ckb-daemon/input_mac.c
src/ckb-daemon/main.c
src/ckb-daemon/profile.c
src/ckb-daemon/usb.c
src/ckb-daemon/usb_linux.c
src/ckb-daemon/usb_mac.c

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)

  os_inputopen
- 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)

5.17.1 Macro Definition Documentation

5.17.1.1 #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().

5.17.2 Function Documentation

5.17.2.1 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.

```
                                                                                  {
    if(keyindex >= N_KEYS_INPUT)
        return;
    // Find the key to bind to
    int tocode = 0;
    if(sscanf(to, "#x%ux", &tocode) != 1 && sscanf(to, "#%u", &tocode) == 1 && tocode <
  N_KEYS_INPUT){
        pthread_mutex_lock(imutex(kb));
        mode->bind.base[keyindex] = tocode;
        pthread_mutex_unlock(imutex(kb));
        return;
    }
    // If not numeric, look it up
    for(int i = 0; i < N_KEYS_INPUT; i++){
        if(keymap[i].name && !strcmp(to, keymap[i].name)){
            pthread_mutex_lock(imutex(kb));
            mode->bind.base[keyindex] = keymap[i].scan;
            pthread_mutex_unlock(imutex(kb));
            return;
        }
    }
}
```

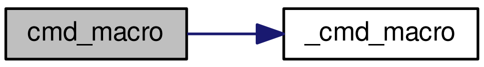
5.17.2.2 void cmd_macro (**usbdevice** ∗ *kb,* **usbmode** ∗ *mode,* const int *notifynumber,* const char ∗ *keys,* const char ∗ *assignment*)

Definition at line 332 of file input.c.

References _cmd_macro(), and imutex.

```
     {
    pthread_mutex_lock(imutex(kb));
    _cmd_macro(mode, keys, assignment);
    pthread_mutex_unlock(imutex(kb));
}
```

Here is the call graph for this function:



5.17.2.3 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.

```
                                                                                 {
    if(keyindex >= N_KEYS_INPUT)
        return;
    pthread_mutex_lock(imutex(kb));
    mode->bind.base[keyindex] = keymap[keyindex].scan;
    pthread_mutex_unlock(imutex(kb));
}
```

5.17.2.4 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.

```
                                                                                 {
    if(keyindex >= N_KEYS_INPUT)
        return;
    pthread_mutex_lock(imutex(kb));
    mode->bind.base[keyindex] = KEY_UNBOUND;
    pthread_mutex_unlock(imutex(kb));
}
```

5.17.2.5 void freebind (binding ∗ *bind*)

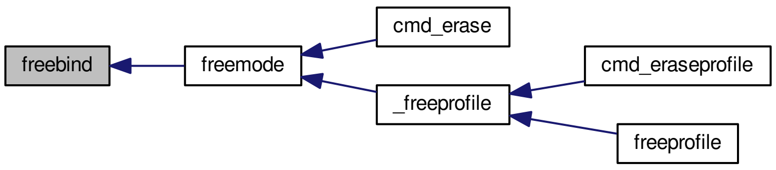
Definition at line 181 of file input.c.

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

Referenced by freemode().

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

Here is the caller graph for this function:



5.17.2.6 void initbind (binding ∗ *bind*)

Definition at line 173 of file input.c.

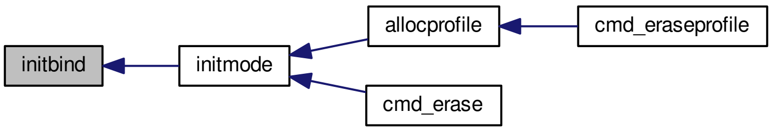
References binding::base, keymap, binding::macrocap, binding::macrocount, binding::macros, N_KEYS_INPUT, and key::scan.

Referenced by initmode().

```
                            {
    for(int i = 0; i < N_KEYS_INPUT; i++)
```

```
        bind->base[i] = keymap[i].scan;
    bind->macros = calloc(32, sizeof(keymacro));
    bind->macrocap = 32;
    bind->macrocount = 0;
}
```

Here is the caller graph for this function:



5.17.2.7 void inputupdate (**usbdevice** ∗ *kb*)

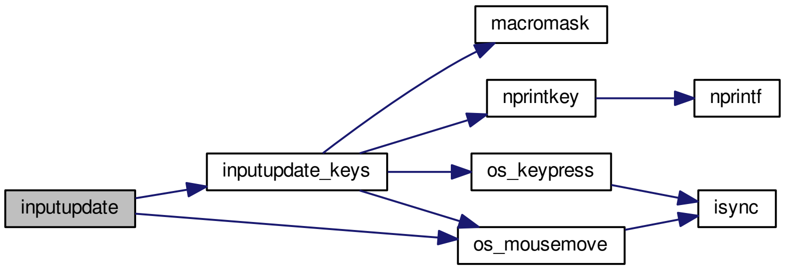
Definition at line 122 of file input.c.

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

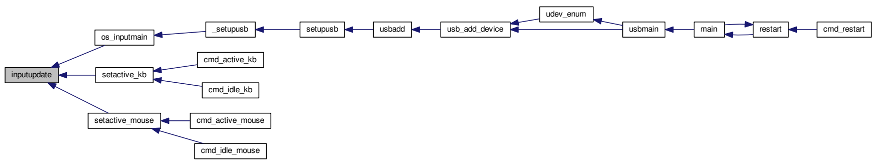
Referenced by os_inputmain(), setactive_kb(), and setactive_mouse().

```
                               {
#ifdef OS_LINUX
    if((!kb->uinput_kb || !kb->uinput_mouse)
#else
    if(!kb->event
#endif
            || !kb->profile)
        return;
    // Process key/button input
    inputupdate_keys(kb);
    // Process mouse movement
    usbinput* input = &kb->input;
    if(input->rel_x != 0 || input->rel_y != 0){
        os_mousemove(kb, input->rel_x, input->rel_y);
        input->rel_x = input->rel_y = 0;
    }
    // Finish up
    memcpy(input->prevkeys, input->keys, N_KEYBYTES_INPUT);
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.17.2.8 void os_inputclose (usbdevice ∗ *kb*)

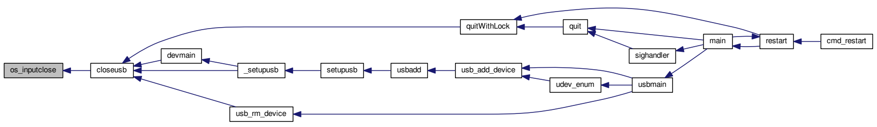
Definition at line 76 of file input_linux.c.

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

Referenced by closeusb().

```
76                                  {
77      if(kb->uinput_kb <= 0 || kb->uinput_mouse <= 0)
78          return;
79      // Set all keys released
80      struct input_event event;
81      memset(&event, 0, sizeof(event));
82      event.type = EV_KEY;
83      for(int key = 0; key < KEY_CNT; key++){
84          event.code = key;
85          if(write(kb->uinput_kb - 1, &event, sizeof(event)) <= 0)
86              ckb_warn("uinput write failed: %s\n", strerror(errno));
87          if(write(kb->uinput_mouse - 1, &event, sizeof(event)) <= 0)
88              ckb_warn("uinput write failed: %s\n", strerror(errno));
89      }
90      event.type = EV_SYN;
91      event.code = SYN_REPORT;
92      if(write(kb->uinput_kb - 1, &event, sizeof(event)) <= 0)
93          ckb_warn("uinput write failed: %s\n", strerror(errno));
94      if(write(kb->uinput_mouse - 1, &event, sizeof(event)) <= 0)
95          ckb_warn("uinput write failed: %s\n", strerror(errno));
96      // Close the keyboard
97      ioctl(kb->uinput_kb - 1, UI_DEV_DESTROY);
98      close(kb->uinput_kb - 1);
99      kb->uinput_kb = 0;
100      // Close the mouse
101      ioctl(kb->uinput_mouse - 1, UI_DEV_DESTROY);
102      close(kb->uinput_mouse - 1);
103      kb->uinput_mouse = 0;
104 }
```

Here is the caller graph for this function:



5.17.2.9 int os_inputopen (usbdevice ∗ *kb*)

Parameters

kb	

Returns

Some tips on using `uinput_user_dev in`

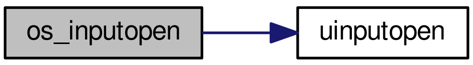
Definition at line 55 of file input_linux.c.

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

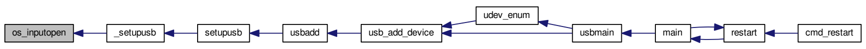
Referenced by _setupusb().

```
                                {
    // Create the new input device
    int index = INDEX_OF(kb, keyboard);
    struct uinput_user_dev indev;
    memset(&indev, 0, sizeof(indev));
    snprintf(indev.name, UINPUT_MAX_NAME_SIZE, "ckb%d: %s", index, kb->name);
    indev.id.bustype = BUS_USB;
    indev.id.vendor = kb->vendor;
    indev.id.product = kb->product;
    indev.id.version = kb->fwversion;
    // Open keyboard
    int fd = uinputopen(&indev, 0);
    kb->uinput_kb = fd;
    if(fd <= 0)
        return 0;
    // Open mouse
    fd = uinputopen(&indev, 1);
    kb->uinput_mouse = fd;
    return fd <= 0;
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.17.2.10 void os_keypress (usbdevice ∗ *kb,* int *scancode,* int *down*)

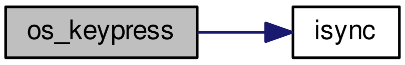
Definition at line 118 of file input_linux.c.

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

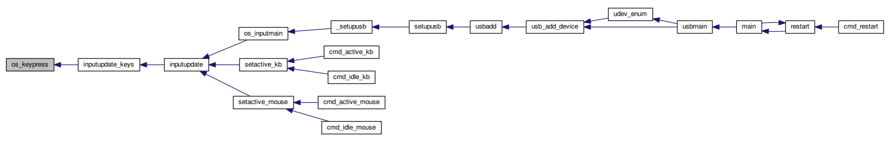
Referenced by inputupdate_keys().

```
118                                                        {
119     struct input_event event;
120     memset(&event, 0, sizeof(event));
121     int is_mouse = 0;
122     if(scancode == BTN_WHEELUP || scancode == BTN_WHEELDOWN){
123         // The mouse wheel is a relative axis
124         if(!down)
125             return;
126         event.type = EV_REL;
127         event.code = REL_WHEEL;
128         event.value = (scancode == BTN_WHEELUP ? 1 : -1);
129         is_mouse = 1;
130     } else {
131         // Mouse buttons and key events are both EV_KEY. The scancodes are already correct, just remove the
       ckb bit
132         event.type = EV_KEY;
133         event.code = scancode & ~SCAN_MOUSE;
134         event.value = down;
135         is_mouse = !!(scancode & SCAN_MOUSE);
136     }
137     if(write((is_mouse ? kb->uinput_mouse : kb->uinput_kb) - 1, &event, sizeof(event))
       <= 0)
138         ckb_warn("uinput write failed: %s\n", strerror(errno));
139     else
140         isync(kb);
141 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.17.2.11 void os_mousemove (usbdevice ∗ *kb,* int *x,* int *y*)

Definition at line 143 of file input_linux.c.

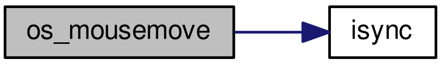
References ckb_warn, isync(), and usbdevice::uinput_mouse.

Referenced by inputupdate(), and inputupdate_keys().

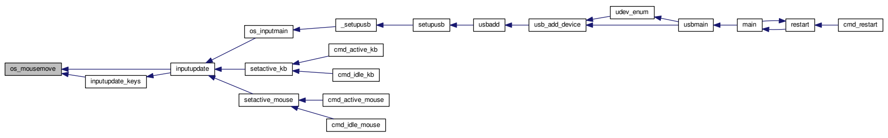
```
143                                               {
144     struct input_event event;
145     memset(&event, 0, sizeof(event));
146     event.type = EV_REL;
147     if(x != 0){
148         event.code = REL_X;
149         event.value = x;
150         if(write(kb->uinput_mouse - 1, &event, sizeof(event))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

```
    if(y != 0){
        event.code = REL_Y;
        event.value = y;
        if(write(kb->uinput_mouse - 1, &event, sizeof(event)) <= 0)
            ckb_warn("uinput write failed: %s\n", strerror(errno));
        else
            isync(kb);
    }
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.17.2.12 int os_setupindicators (**usbdevice** ∗ ***kb***)

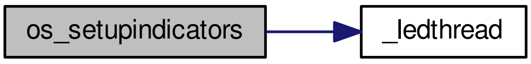
Definition at line 189 of file input_linux.c.

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

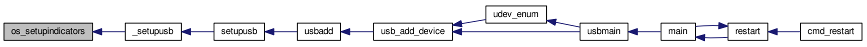
Referenced by _setupusb().

```
                                        {
    // Initialize LEDs to all off
    kb->hw_ileds = kb->hw_ileds_old = kb->ileds = 0;
    // Create and detach thread to read LED events
    pthread_t thread;
    int err = pthread_create(&thread, 0, _ledthread, kb);
    if(err != 0)
        return err;
    pthread_detach(thread);
    return 0;
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



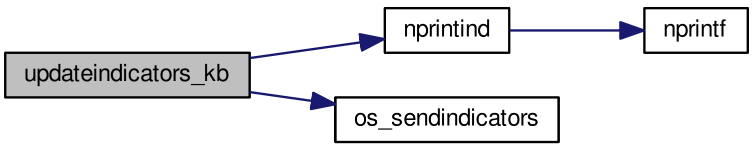
5.17.2.13 void updateindicators_kb (**usbdevice** ∗ *kb,* int *force*)

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 }
```

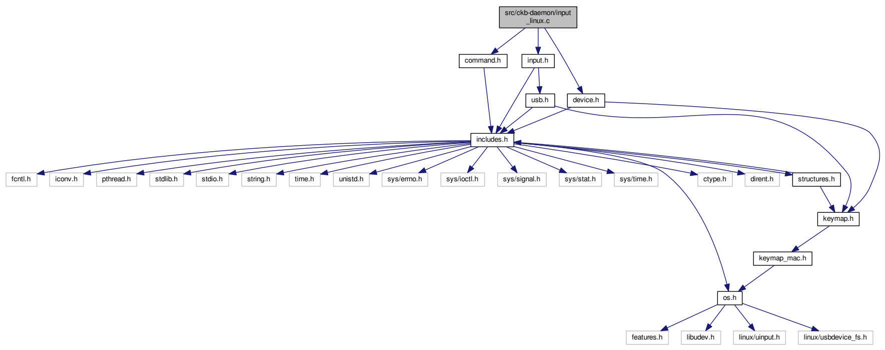
Here is the call graph for this function:



5.18 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)

  os_inputopen
- 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)

5.18.1 Function Documentation

5.18.1.1 void∗ _ledthread (void ∗ *ctx*)

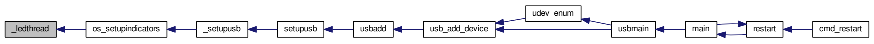
Definition at line 165 of file input_linux.c.

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

Referenced by os_setupindicators().

```
165                            {
166     usbdevice* kb = ctx;
167     uchar ileds = 0;
168     // Read LED events from the uinput device
169     struct input_event event;
170     while (read(kb->uinput_kb - 1, &event, sizeof(event)) > 0) {
171         if (event.type == EV_LED && event.code < 8){
172             char which = 1 << event.code;
173             if(event.value)
174                 ileds |= which;
175             else
176                 ileds &= ~which;
177         }
178         // Update them if needed
179         pthread_mutex_lock(dmutex(kb));
180         if(kb->hw_ileds != ileds){
181             kb->hw_ileds = ileds;
182             kb->vtable->updateindicators(kb, 0);
183         }
184         pthread_mutex_unlock(dmutex(kb));
185     }
186     return 0;
187 }
```

Here is the caller graph for this function:



5.18.1.2 static void isync (**usbdevice** ∗ *kb*) [static]

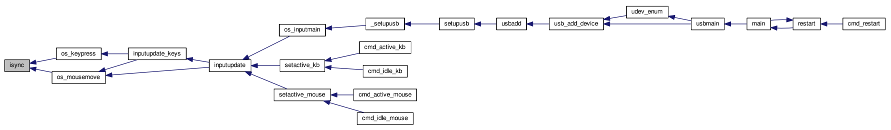
Definition at line 107 of file input_linux.c.

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

Referenced by os_keypress(), and os_mousemove().

```
107                                  {
108     struct input_event event;
109     memset(&event, 0, sizeof(event));
110     event.type = EV_SYN;
111     event.code = SYN_REPORT;
112     if(write(kb->uinput_kb - 1, &event, sizeof(event)) <= 0)
113         ckb_warn("uinput write failed: %s\n", strerror(errno));
114     if(write(kb->uinput_mouse - 1, &event, sizeof(event)) <= 0)
115         ckb_warn("uinput write failed: %s\n", strerror(errno));
116 }
```

Here is the caller graph for this function:



5.18.1.3 void os_inputclose (**usbdevice** ∗ *kb*)

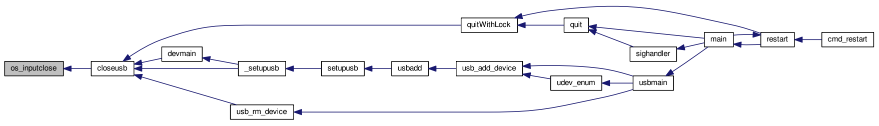
Definition at line 76 of file input_linux.c.

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

Referenced by closeusb().

```
76                                 {
77     if(kb->uinput_kb <= 0 || kb->uinput_mouse <= 0)
78         return;
79     // Set all keys released
80     struct input_event event;
81     memset(&event, 0, sizeof(event));
82     event.type = EV_KEY;
83     for(int key = 0; key < KEY_CNT; key++){
84         event.code = key;
85         if(write(kb->uinput_kb - 1, &event, sizeof(event)) <= 0)
86             ckb_warn("uinput write failed: %s\n", strerror(errno));
87         if(write(kb->uinput_mouse - 1, &event, sizeof(event)) <= 0)
88             ckb_warn("uinput write failed: %s\n", strerror(errno));
89     }
90     event.type = EV_SYN;
91     event.code = SYN_REPORT;
92     if(write(kb->uinput_kb - 1, &event, sizeof(event)) <= 0)
93         ckb_warn("uinput write failed: %s\n", strerror(errno));
94     if(write(kb->uinput_mouse - 1, &event, sizeof(event)) <= 0)
95         ckb_warn("uinput write failed: %s\n", strerror(errno));
96     // Close the keyboard
97     ioctl(kb->uinput_kb - 1, UI_DEV_DESTROY);
98     close(kb->uinput_kb - 1);
99     kb->uinput_kb = 0;
100     // Close the mouse
101     ioctl(kb->uinput_mouse - 1, UI_DEV_DESTROY);
102     close(kb->uinput_mouse - 1);
103     kb->uinput_mouse = 0;
104 }
```

Here is the caller graph for this function:



5.18.1.4 int os_inputopen (**usbdevice** ∗ *kb*)

Parameters

kb	

Returns

Some tips on using `uinput_user_dev in`

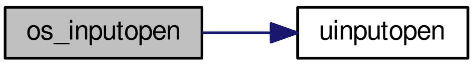
Definition at line 55 of file input_linux.c.

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

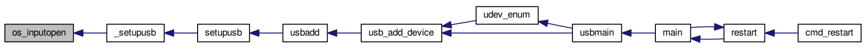
Referenced by _setupusb().

```
55                              {
56     // Create the new input device
57     int index = INDEX_OF(kb, keyboard);
58     struct uinput_user_dev indev;
59     memset(&indev, 0, sizeof(indev));
60     snprintf(indev.name, UINPUT_MAX_NAME_SIZE, "ckb%d: %s", index, kb->name);
61     indev.id.bustype = BUS_USB;
62     indev.id.vendor = kb->vendor;
63     indev.id.product = kb->product;
64     indev.id.version = kb->fwversion;
65     // Open keyboard
66     int fd = uinputopen(&indev, 0);
67     kb->uinput_kb = fd;
68     if(fd <= 0)
69         return 0;
70     // Open mouse
71     fd = uinputopen(&indev, 1);
72     kb->uinput_mouse = fd;
73     return fd <= 0;
74 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.18.1.5 void os_keypress (**usbdevice** ∗ *kb,* int *scancode,* int *down*)

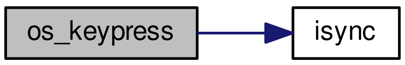
Definition at line 118 of file input_linux.c.

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

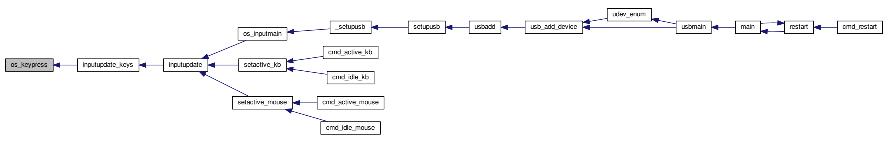
Referenced by inputupdate_keys().

```
118                                                         {
119     struct input_event event;
120     memset(&event, 0, sizeof(event));
121     int is_mouse = 0;
122     if(scancode == BTN_WHEELUP || scancode == BTN_WHEELDOWN){
123         // The mouse wheel is a relative axis
124         if(!down)
125             return;
126         event.type = EV_REL;
127         event.code = REL_WHEEL;
128         event.value = (scancode == BTN_WHEELUP ? 1 : -1);
129         is_mouse = 1;
130     } else {
131         // Mouse buttons and key events are both EV_KEY. The scancodes are already correct, just remove the
       ckb bit
132         event.type = EV_KEY;
133         event.code = scancode & ~SCAN_MOUSE;
134         event.value = down;
135         is_mouse = !!(scancode & SCAN_MOUSE);
136     }
137     if(write((is_mouse ? kb->uinput_mouse : kb->uinput_kb) - 1, &event, sizeof(event))
       <= 0)
138         ckb_warn("uinput write failed: %s\n", strerror(errno));
139     else
140         isync(kb);
141 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.18.1.6 void os_mousemove (**usbdevice** ∗ *kb,* int *x,* int *y*)

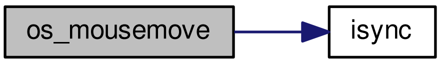
Definition at line 143 of file input_linux.c.

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

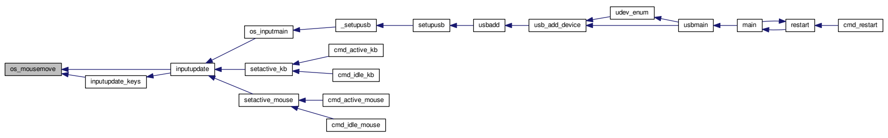
Referenced by inputupdate(), and inputupdate_keys().

```
143                                               {
144     struct input_event event;
145     memset(&event, 0, sizeof(event));
146     event.type = EV_REL;
147     if(x != 0){
148         event.code = REL_X;
149         event.value = x;
150         if(write(kb->uinput_mouse - 1, &event, sizeof(event)) <= 0)
151             ckb_warn("uinput write failed: %s\n", strerror(errno));
152         else
153             isync(kb);
154     }
155     if(y != 0){
156         event.code = REL_Y;
157         event.value = y;
158         if(write(kb->uinput_mouse - 1, &event, sizeof(event)) <= 0)
159             ckb_warn("uinput write failed: %s\n", strerror(errno));
160         else
161             isync(kb);
162     }
163 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.18.1.7 int os_setupindicators (**usbdevice** ∗ *kb*)

Definition at line 189 of file input_linux.c.

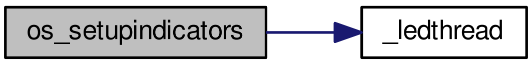
References _ledthread(), usbdevice::hw_ileds, usbdevice::hw_ileds_old, and usbdevice::ileds.

Referenced by _setupusb().

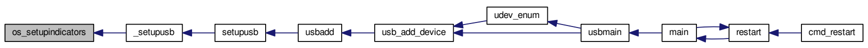
```
189                                       {
190     // Initialize LEDs to all off
191     kb->hw_ileds = kb->hw_ileds_old = kb->ileds = 0;
192     // Create and detach thread to read LED events
193     pthread_t thread;
194     int err = pthread_create(&thread, 0, _ledthread, kb);
195     if(err != 0)
```

```
        return err;
    pthread_detach(thread);
    return 0;
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.18.1.8 int uinputopen (struct uinput_user_dev ∗ *indev,* int *mouse*)

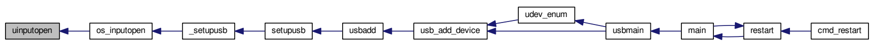
Definition at line 9 of file input_linux.c.

References ckb_err, and ckb_warn.

Referenced by os_inputopen().

```
                                                       {
    int fd = open("/dev/uinput", O_RDWR);
    if(fd < 0){
        // If that didn't work, try /dev/input/uinput instead
        fd = open("/dev/input/uinput", O_RDWR);
        if(fd < 0){
            ckb_err("Failed to open uinput: %s\n", strerror(errno));
            return 0;
        }
    }
    // Enable all keys and mouse buttons
    ioctl(fd, UI_SET_EVBIT, EV_KEY);
    for(int i = 0; i < KEY_CNT; i++)
        ioctl(fd, UI_SET_KEYBIT, i);
    if(mouse){
        // Enable mouse axes
        ioctl(fd, UI_SET_EVBIT, EV_REL);
        for(int i = 0; i < REL_CNT; i++)
            ioctl(fd, UI_SET_RELBIT, i);
    } else {
        // Enable LEDs
        ioctl(fd, UI_SET_EVBIT, EV_LED);
        for(int i = 0; i < LED_CNT; i++)
            ioctl(fd, UI_SET_LEDBIT, i);
        // Eanble autorepeat
        ioctl(fd, UI_SET_EVBIT, EV_REP);
    }
    // Enable sychronization
    ioctl(fd, UI_SET_EVBIT, EV_SYN);
    // Create the device
    if(write(fd, indev, sizeof(*indev)) <= 0)
        ckb_warn("uinput write failed: %s\n", strerror(errno));
    if(ioctl(fd, UI_DEV_CREATE)){
        ckb_err("Failed to create uinput device: %s\n", strerror(errno));
        close(fd);
        return 0;
    }
    return fd + 1;
}
```

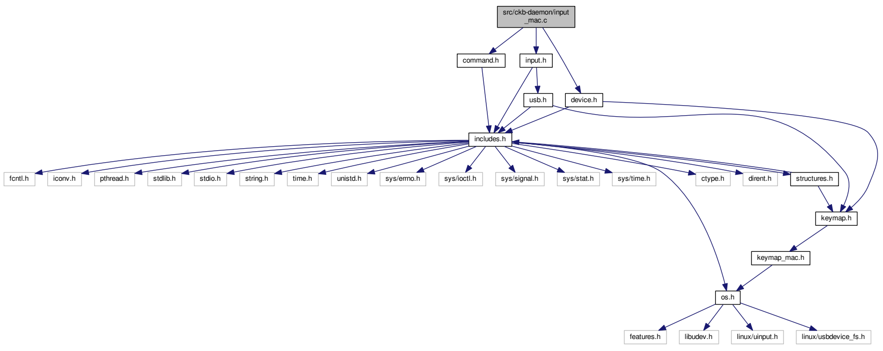
Here is the caller graph for this function:



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

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

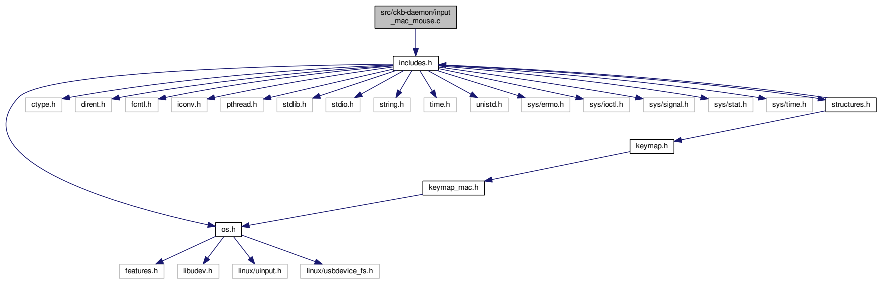
Include dependency graph for input_mac.c:



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

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

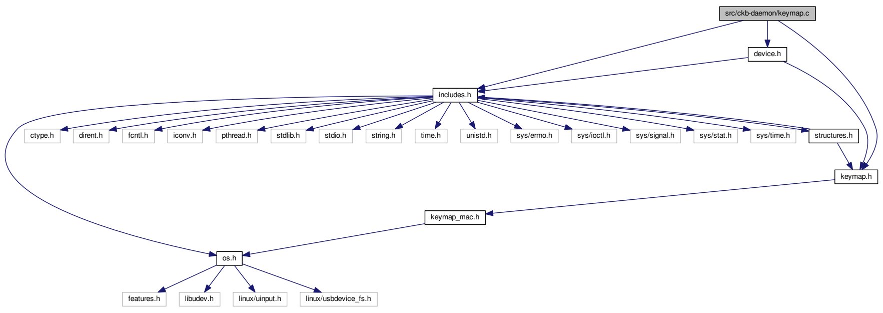
Include dependency graph for input_mac_mouse.c:



5.21 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)]

5.21.1 Macro Definition Documentation

5.21.1.1 #define BUTTON_HID_COUNT 5

Definition at line 364 of file keymap.c.

Referenced by corsair_mousecopy(), and hid_mouse_translate().

5.21.2 Function Documentation

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

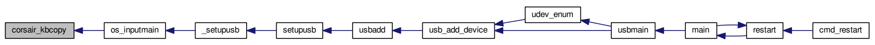
Definition at line 394 of file keymap.c.

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:



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

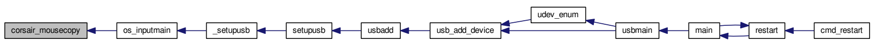
Definition at line 403 of file keymap.c.

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:



5.21.2.3 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.

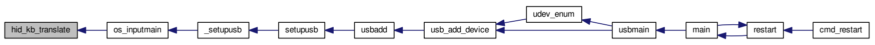
Referenced by os_inputmain().

```
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
```

```
        120, 121, 122, 123, 124, 125, 126, 127, 128, 129, 136, 137, 138, 139, 140, 141,
         60,  48,  62,  61,  91,  90,  67,  68, 142, 143,  99, 101,  -2, 130, 131,  97,
         -2, 133, 134, 135,  -2,  96,  -2, 132,  -2,  -2,  71,  71,  71,  71,  -1,  -1,
    };
    switch(endpoint){
    case 1:
    case -1:
        // EP 1: 6KRO input (RGB and non-RGB)
        // Clear previous input
        for(int i = 0; i < 256; i++){
            if(hid_codes[i] >= 0)
                CLEAR_KEYBIT(kbinput, hid_codes[i]);
        }
        // Set new input
        for(int i = 0; i < 8; i++){
            if((urbinput[0] >> i) & 1)
                SET_KEYBIT(kbinput, hid_codes[i + 224]);
        }
        for(int i = 2; i < length; i++){
            if(urbinput[i] > 3){
                int scan = hid_codes[urbinput[i]];
                if(scan >= 0)
                    SET_KEYBIT(kbinput, scan);
                else
                    ckb_warn("Got unknown key press %d on EP 1\n", urbinput[i]);
            }
        }
        break;
    case -2:
        // EP 2 RGB: NKRO input
        if(urbinput[0] == 1){
            // Type 1: standard key
            if(length != 21)
                return;
            for(int bit = 0; bit < 8; bit++){
                if((urbinput[1] >> bit) & 1)
                    SET_KEYBIT(kbinput, hid_codes[bit + 224]);
                else
                    CLEAR_KEYBIT(kbinput, hid_codes[bit + 224]);
            }
            for(int byte = 0; byte < 19; byte++){
                char input = urbinput[byte + 2];
                for(int bit = 0; bit < 8; bit++){
                    int keybit = byte * 8 + bit;
                    int scan = hid_codes[keybit];
                    if((input >> bit) & 1){
                        if(scan >= 0)
                            SET_KEYBIT(kbinput, hid_codes[keybit]);
                        else
                            ckb_warn("Got unknown key press %d on EP 2\n", keybit);
                    } else if(scan >= 0)
                        CLEAR_KEYBIT(kbinput, hid_codes[keybit]);
                }
            }
            break;
        } else if(urbinput[0] == 2)
            ;        // Type 2: media key (fall through)
        else
            break;  // No other known types
    case 2:
        // EP 2 Non-RGB: media keys
        CLEAR_KEYBIT(kbinput, 97);          // mute
        CLEAR_KEYBIT(kbinput, 98);          // stop
        CLEAR_KEYBIT(kbinput, 99);          // prev
        CLEAR_KEYBIT(kbinput, 100);         // play
        CLEAR_KEYBIT(kbinput, 101);         // next
        CLEAR_KEYBIT(kbinput, 130);         // volup
        CLEAR_KEYBIT(kbinput, 131);         // voldn
        for(int i = 0; i < length; i++){
            switch(urbinput[i]){
            case 181:
                SET_KEYBIT(kbinput, 101);   // next
                break;
            case 182:
                SET_KEYBIT(kbinput, 99);    // prev
                break;
            case 183:
                SET_KEYBIT(kbinput, 98);    // stop
                break;
            case 205:
                SET_KEYBIT(kbinput, 100);   // play
                break;
            case 226:
                SET_KEYBIT(kbinput, 97);    // mute
                break;
            case 233:
                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:



5.21.2.4 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.

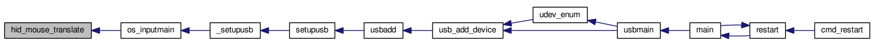
References BUTTON_HID_COUNT, CLEAR_KEYBIT, MOUSE_BUTTON_FIRST, MOUSE_EXTRA_FIRST, and S-ET_KEYBIT.

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)
```

```
        SET_KEYBIT(kbinput, MOUSE_EXTRA_FIRST + 1);      // wheeldn
    else
        CLEAR_KEYBIT(kbinput, MOUSE_EXTRA_FIRST + 1);
}
```

Here is the caller graph for this function:



5.21.3 Variable Documentation

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

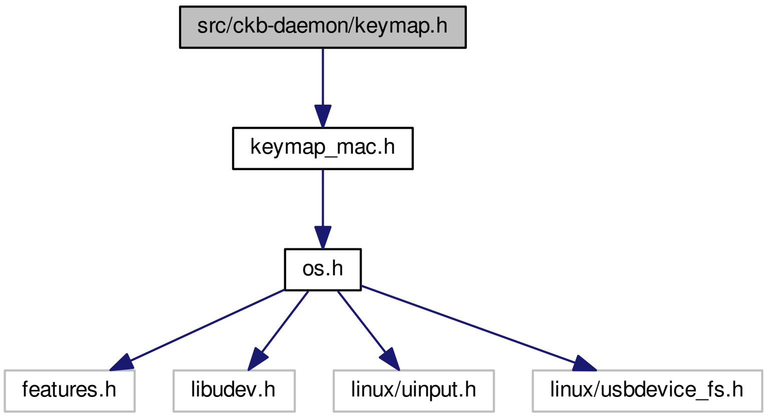
Definition at line 5 of file keymap.c.

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

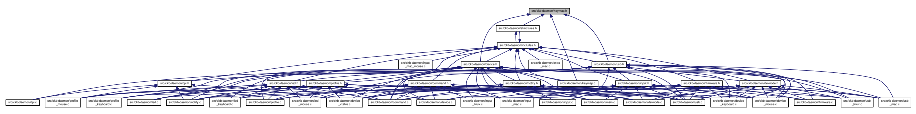
5.22 src/ckb-daemon/keymap.h File Reference

```
#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

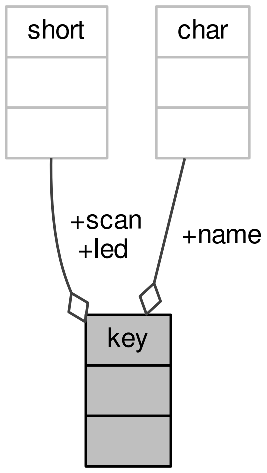
- const key keymap [(((152+3+12)+25)+11)]

5.22.1 Data Structure Documentation

5.22.1.1 struct key

Definition at line 49 of file keymap.h.

Collaboration diagram for key:



Data Fields

short	led	
const char ∗	name	
short	scan	

5.22.2 Macro Definition Documentation

5.22.2.1 #define BTN_WHEELDOWN 0x1f02

Definition at line 13 of file keymap.h.

Referenced by os_keypress().

5.22.2.2 #define BTN_WHEELUP 0x1f01

Definition at line 12 of file keymap.h.

Referenced by os_keypress().

5.22.2.3 #define KEY_BACKSLASH_ISO KEY_BACKSLASH

Definition at line 20 of file keymap.h.

5.22.2.4 #define KEY_CORSAIR -2

Definition at line 8 of file keymap.h.

5.22.2.5 #define KEY_NONE -1

Definition at line 7 of file keymap.h.

5.22.2.6 #define KEY_UNBOUND -3

Definition at line 9 of file keymap.h.

Referenced by cmd_unbind().

5.22.2.7 #define LED_DPI (LED_MOUSE + 2)

Definition at line 43 of file keymap.h.

Referenced by loadrgb_mouse(), and savergb_mouse().

5.22.2.8 #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().

5.22.2.9 #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().

5.22.2.10 #define MOUSE_EXTRA_FIRST (MOUSE_BUTTON_FIRST + N_BUTTONS_HW)

Definition at line 34 of file keymap.h.

Referenced by hid_mouse_translate().

5.22.2.11 #define N_BUTTONS_EXTENDED 25

Definition at line 32 of file keymap.h.

5.22.2.12 #define N_BUTTONS_HW 20

Definition at line 31 of file keymap.h.

Referenced by corsair_mousecopy().

5.22.2.13 #define N_KEY_ZONES 3

Definition at line 27 of file keymap.h.

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

Definition at line 46 of file keymap.h.

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

Definition at line 25 of file keymap.h.

Referenced by corsair_kbcopy().

5.22.2.16 #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().

5.22.2.17 #define N_KEYS_EXTENDED (N_KEYS_INPUT + N_MOUSE_ZONES_EXTENDED)

Definition at line 45 of file keymap.h.

Referenced by printrgb(), and readcmd().

5.22.2.18 #define N_KEYS_EXTRA 12

Definition at line 29 of file keymap.h.

5.22.2.19 #define N_KEYS_HW 152

Definition at line 24 of file keymap.h.

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

5.22.2.20 #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().

5.22.2.21 #define N_MOUSE_ZONES 5

Definition at line 40 of file keymap.h.

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

5.22.2.22 #define N_MOUSE_ZONES_EXTENDED 11

Definition at line 41 of file keymap.h.

5.22.2.23 #define SCAN_KBD 0

Definition at line 57 of file keymap.h.

5.22.2.24 #define SCAN_MOUSE 0x1000

Definition at line 58 of file keymap.h.

Referenced by os_keypress().

5.22.2.25 #define SCAN_SILENT 0x8000

Definition at line 56 of file keymap.h.

Referenced by inputupdate_keys().

5.22.3 Function Documentation

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

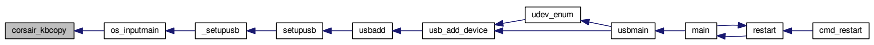
Definition at line 394 of file keymap.c.

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:



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

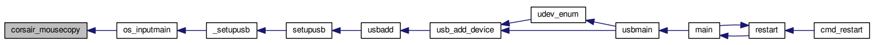
Definition at line 403 of file keymap.c.

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:



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

Definition at line 223 of file keymap.c.

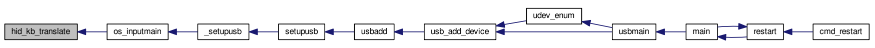
References ckb_warn, CLEAR_KEYBIT, and SET_KEYBIT.

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)
```

```
            ;       // Type 2: media key (fall through)
        else
            break;  // No other known types
    case 2:
        // EP 2 Non-RGB: media keys
        CLEAR_KEYBIT(kbinput, 97);          // mute
        CLEAR_KEYBIT(kbinput, 98);          // stop
        CLEAR_KEYBIT(kbinput, 99);          // prev
        CLEAR_KEYBIT(kbinput, 100);         // play
        CLEAR_KEYBIT(kbinput, 101);         // next
        CLEAR_KEYBIT(kbinput, 130);         // volup
        CLEAR_KEYBIT(kbinput, 131);         // voldn
        for(int i = 0; i < length; i++){
            switch(urbinput[i]){
            case 181:
                SET_KEYBIT(kbinput, 101);   // next
                break;
            case 182:
                SET_KEYBIT(kbinput, 99);    // prev
                break;
            case 183:
                SET_KEYBIT(kbinput, 98);    // stop
                break;
            case 205:
                SET_KEYBIT(kbinput, 100);   // play
                break;
            case 226:
                SET_KEYBIT(kbinput, 97);    // mute
                break;
            case 233:
                SET_KEYBIT(kbinput, 130);   // volup
                break;
            case 234:
                SET_KEYBIT(kbinput, 131);   // voldn
                break;
            }
        }
        break;
    case 3:
        // EP 3 non-RGB: NKRO input
        if(length != 15)
            return;
        for(int bit = 0; bit < 8; bit++){
            if((urbinput[0] >> bit) & 1)
                SET_KEYBIT(kbinput, hid_codes[bit + 224]);
            else
                CLEAR_KEYBIT(kbinput, hid_codes[bit + 224]);
        }
        for(int byte = 0; byte < 14; byte++){
            char input = urbinput[byte + 1];
            for(int bit = 0; bit < 8; bit++){
                int keybit = byte * 8 + bit;
                int scan = hid_codes[keybit];
                if((input >> bit) & 1){
                    if(scan >= 0)
                        SET_KEYBIT(kbinput, hid_codes[keybit]);
                    else
                        ckb_warn("Got unknown key press %d on EP 3\n", keybit);
                } else if(scan >= 0)
                    CLEAR_KEYBIT(kbinput, hid_codes[keybit]);
            }
        }
        break;
    }
}
```

Here is the caller graph for this function:



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

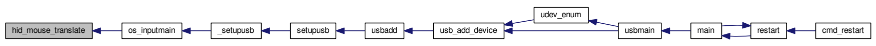
Definition at line 366 of file keymap.c.

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

Referenced by os_inputmain().

```
                                    {
    if((endpoint != 2 && endpoint != -2) || length < 10)
        return;
    // EP 2: mouse input
    if(urbinput[0] != 1)
        return;
    // Byte 1 = mouse buttons (bitfield)
    for(int bit = 0; bit < BUTTON_HID_COUNT; bit++){
        if(urbinput[1] & (1 << bit))
            SET_KEYBIT(kbinput, MOUSE_BUTTON_FIRST + bit);
        else
            CLEAR_KEYBIT(kbinput, MOUSE_BUTTON_FIRST + bit);
    }
    // Bytes 5 - 8: movement
    *xaxis += *(short*)(urbinput + 5);
    *yaxis += *(short*)(urbinput + 7);
    // Byte 9: wheel
    char wheel = urbinput[9];
    if(wheel > 0)
        SET_KEYBIT(kbinput, MOUSE_EXTRA_FIRST);          // wheelup
    else
        CLEAR_KEYBIT(kbinput, MOUSE_EXTRA_FIRST);
    if(wheel < 0)
        SET_KEYBIT(kbinput, MOUSE_EXTRA_FIRST + 1);      // wheeldn
    else
        CLEAR_KEYBIT(kbinput, MOUSE_EXTRA_FIRST + 1);
}
```

Here is the caller graph for this function:



5.22.4 Variable Documentation

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

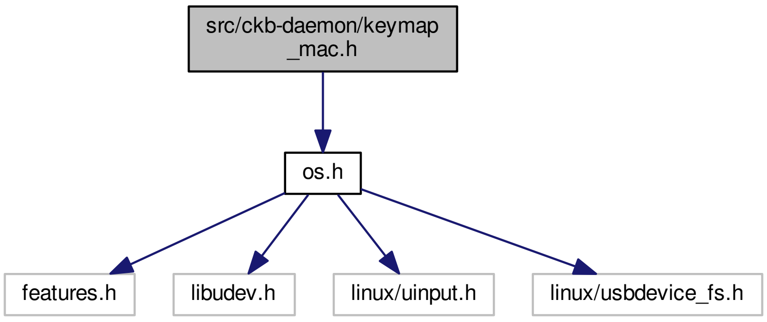
Definition at line 5 of file keymap.c.

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

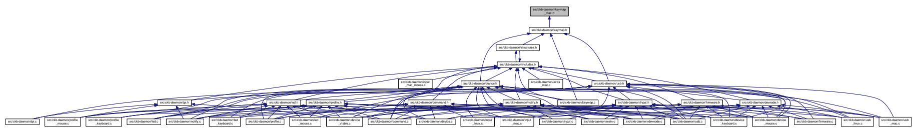
5.23 src/ckb-daemon/keymap_mac.h File Reference

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

Include dependency graph for keymap_mac.h:



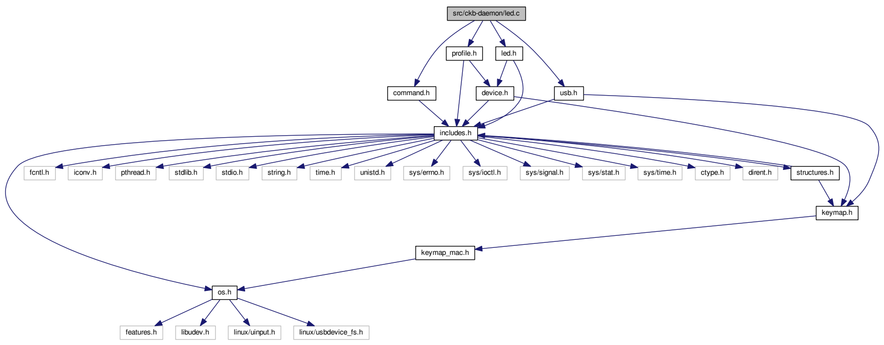
This graph shows which files directly or indirectly include this file:



5.24 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)

5.24.1 Function Documentation

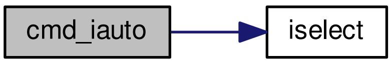
5.24.1.1 void cmd_iauto (**usbdevice** ∗ *kb,* **usbmode** ∗ *mode,* int *dummy1,* int *dummy2,* const char ∗ *led*)

Definition at line 54 of file led.c.

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

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

Here is the call graph for this function:



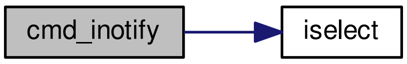
5.24.1.2 void cmd_inotify (**usbdevice** ∗ *kb,* **usbmode** ∗ *mode,* int *nnumber,* int *dummy,* const char ∗ *led*)

Definition at line 62 of file led.c.

References usbmode::inotify, and iselect().

```
                                                                                    {
    uchar bits = iselect(led);
    if(strstr(led, ":off"))
        // Turn notifications for these bits off
        mode->inotify[nnumber] &= ~bits;
    else
        // Turn notifications for these bits on
        mode->inotify[nnumber] |= bits;
}
```

Here is the call graph for this function:



5.24.1.3 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.

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

Here is the call graph for this function:

cmd_ioff
iselect

5.24.1.4 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.

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

Here is the call graph for this function:

cmd_ion
iselect

5.24.1.5 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.

```
                                                                                  {
    int index = keymap[keyindex].led;
    if(index < 0) {
        if (index == -2){      // Process strafe sidelights
            uchar sideshine;
            if (sscanf(code, "%2hhx",&sideshine)) // monochromatic
                mode->light.sidelight = sideshine;
        }
        return;
    }
    uchar r, g, b;
    if(sscanf(code, "%2hhx%2hhx%2hhx", &r, &g, &b) == 3){
        mode->light.r[index] = r;
        mode->light.g[index] = g;
        mode->light.b[index] = b;
    }
}
```

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

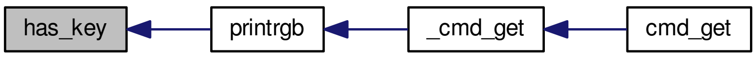
Definition at line 73 of file led.c.

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

Referenced by printrgb().

```
                                                                {
    if(!name)
        return 0;
    if(IS_MOUSE(kb->vendor, kb->product)){
        // Mice only have the RGB zones
        if((IS_SABRE(kb) || IS_SCIMITAR(kb)) && !strcmp(name, "wheel"))
            return 1;
        if(IS_SCIMITAR(kb) && !strcmp(name, "thumb"))
            return 1;
        if(strstr(name, "dpi") == name || !strcmp(name, "front") || !strcmp(name, "back"))
            return 1;
        return 0;
    } else {
        // But keyboards don't have them at all
        if(strstr(name, "dpi") == name || !strcmp(name, "front") || !strcmp(name, "back") || !strcmp(name,
      "wheel") || !strcmp(name, "thumb"))
            return 0;
        // Only K95 has G keys and M keys (G1 - G18, MR, M1 - M3)
        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'))))
            return 0;
        // Only K65 has lights on VolUp/VolDn
        if(!IS_K65(kb) && (!strcmp(name, "volup") || !strcmp(name, "voldn")))
            return 0;
        // K65 lacks numpad and media buttons
        if(IS_K65(kb) && (strstr(name, "num") == name || !strcmp(name, "stop") || !strcmp(name, "prev
      ") || !strcmp(name, "play") || !strcmp(name, "next")))
            return 0;
    }
    return 1;
}
```

Here is the caller graph for this function:



5.24.1.7 static uchar iselect (const char ∗ *led*) [static]

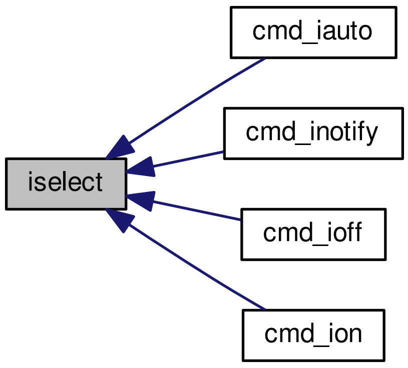
Definition at line 25 of file led.c.

References I_CAPS, I_NUM, and I_SCROLL.

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

```
                                    {
    int result = 0;
    if(!strncmp(led, "num", 3) || strstr(led, ",num"))
        result |= I_NUM;
    if(!strncmp(led, "caps", 4) || strstr(led, ",caps"))
        result |= I_CAPS;
    if(!strncmp(led, "scroll", 6) || strstr(led, ",scroll"))
        result |= I_SCROLL;
    if(!strncmp(led, "all", 3) || strstr(led, ",all"))
        result |= I_NUM | I_CAPS | I_SCROLL;
    return result;
}
```

Here is the caller graph for this function:



5.24.1.8 char∗ printrgb (const lighting ∗ *light*, const usbdevice ∗ *kb*)

Definition at line 102 of file led.c.

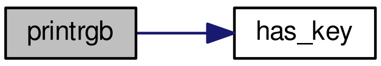
References lighting::b, lighting::g, has_key(), keymap, key::led, N_KEYS_EXTENDED, key::name, and lighting::r.

Referenced by _cmd_get().

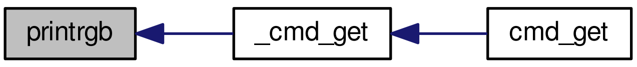
```
                                                        {
    uchar r[N_KEYS_EXTENDED], g[N_KEYS_EXTENDED], b[
  N_KEYS_EXTENDED];
    const uchar* mr = light->r;
    const uchar* mg = light->g;
    const uchar* mb = light->b;
    for(int i = 0; i < N_KEYS_EXTENDED; i++){
        // Translate the key index to an RGB index using the key map
        int k = keymap[i].led;
        if(k < 0)
            continue;
        r[i] = mr[k];
        g[i] = mg[k];
        b[i] = mb[k];
    }
    // Make a buffer to track key names and to filter out duplicates
    char names[N_KEYS_EXTENDED][11];
    for(int i = 0; i < N_KEYS_EXTENDED; i++){
        const char* name = keymap[i].name;
        if(keymap[i].led < 0 || !has_key(name, kb))
            names[i][0] = 0;
        else
```

```
            strncpy(names[i], name, 11);
    }
    // Check to make sure these aren't all the same color
    int same = 1;
    for(int i = 1; i < N_KEYS_EXTENDED; i++){
        if(!names[i][0])
            continue;
        if(r[i] != r[0] || g[i] != g[0] || b[i] != b[0]){
            same = 0;
            break;
        }
    }
    // If they are, just output that color
    if(same){
        char* buffer = malloc(7);
        snprintf(buffer, 7, "%02x%02x%02x", r[0], g[0], b[0]);
        return buffer;
    }
    const int BUFFER_LEN = 4096;    // Should be more than enough to fit all keys
    char* buffer = malloc(BUFFER_LEN);
    int length = 0;
    for(int i = 0; i < N_KEYS_EXTENDED; i++){
        if(!names[i][0])
            continue;
        // Print the key name
        int newlen = 0;
        snprintf(buffer + length, BUFFER_LEN - length, length == 0 ? "%s%n" : " %s%n", names[i], &newlen);
        length += newlen;
        // Look ahead to see if any other keys have this color. If so, print them here as well.
        uchar kr = r[i], kg = g[i], kb = b[i];
        for(int j = i + 1; j < N_KEYS_EXTENDED; j++){
            if(!names[j][0])
                continue;
            if(r[j] != kr || g[j] != kg || b[j] != kb)
                continue;
            snprintf(buffer + length, BUFFER_LEN - length, ",%s%n", names[j], &newlen);
            length += newlen;
            // Erase the key's name so it won't get printed later
            names[j][0] = 0;
        }
        // Print the color
        snprintf(buffer + length, BUFFER_LEN - length, ":%02x%02x%02x%n", kr, kg, kb, &newlen);
        length += newlen;
    }
    return buffer;
}
```

Here is the call graph for this function:



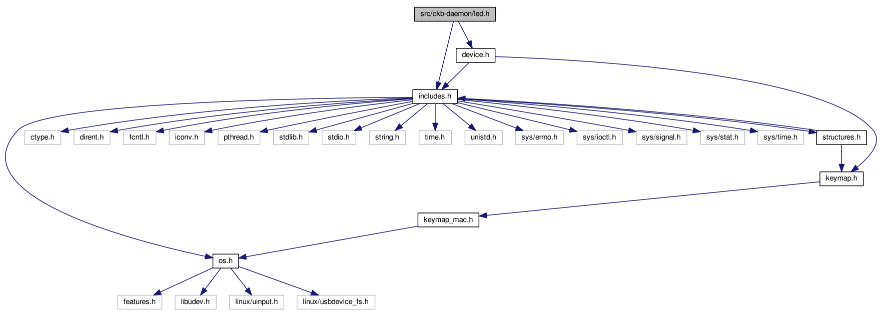
Here is the caller graph for this function:



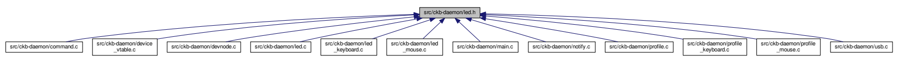
5.25 src/ckb-daemon/led.h File Reference

```
#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)

5.25.1 Function Documentation

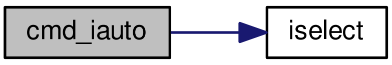
5.25.1.1 void cmd_iauto (**usbdevice** ∗ *kb,* **usbmode** ∗ *mode,* int *dummy1,* int *dummy2,* const char ∗ *led*)

Definition at line 54 of file led.c.

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

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

Here is the call graph for this function:



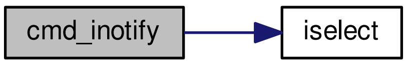
5.25.1.2 void cmd_inotify (usbdevice * *kb*, usbmode * *mode*, int *nnumber*, int *dummy*, const char * *led*)

Definition at line 62 of file led.c.

References usbmode::inotify, and iselect().

```
                                                                                          {
    uchar bits = iselect(led);
    if(strstr(led, ":off"))
        // Turn notifications for these bits off
        mode->inotify[nnumber] &= ~bits;
    else
        // Turn notifications for these bits on
        mode->inotify[nnumber] |= bits;
}
```

Here is the call graph for this function:



5.25.1.3 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.

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

Here is the call graph for this function:

cmd_ioff
iselect

5.25.1.4 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.

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

Here is the call graph for this function:

cmd_ion
iselect

5.25.1.5 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.

```
                                                                                {
    int index = keymap[keyindex].led;
    if(index < 0) {
        if (index == -2){     // Process strafe sidelights
            uchar sideshine;
            if (sscanf(code, "%2hhx",&sideshine)) // monochromatic
                mode->light.sidelight = sideshine;
        }
        return;
    }
    uchar r, g, b;
    if(sscanf(code, "%2hhx%2hhx%2hhx", &r, &g, &b) == 3){
        mode->light.r[index] = r;
        mode->light.g[index] = g;
        mode->light.b[index] = b;
    }
}
```

5.25.1.6 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.

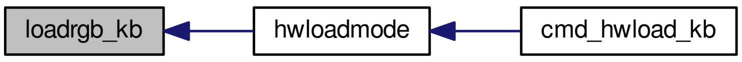
Definition at line 181 of file led_keyboard.c.

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

Referenced by hwloadmode().

```
                                                       {
    if(kb->fwversion >= 0x0120){
        uchar data_pkt[12][MSG_SIZE] = {
            { 0x0e, 0x14, 0x03, 0x01, 0x01, mode + 1, 0x01 },
            { 0xff, 0x01, 60, 0 },
            { 0xff, 0x02, 60, 0 },
            { 0xff, 0x03, 24, 0 },
            { 0x0e, 0x14, 0x03, 0x01, 0x01, mode + 1, 0x02 },
            { 0xff, 0x01, 60, 0 },
            { 0xff, 0x02, 60, 0 },
            { 0xff, 0x03, 24, 0 },
            { 0x0e, 0x14, 0x03, 0x01, 0x01, mode + 1, 0x03 },
            { 0xff, 0x01, 60, 0 },
            { 0xff, 0x02, 60, 0 },
            { 0xff, 0x03, 24, 0 },
        };
        uchar in_pkt[4][MSG_SIZE] = {
            { 0x0e, 0x14, 0x03, 0x01 },
            { 0xff, 0x01, 60, 0 },
            { 0xff, 0x02, 60, 0 },
            { 0xff, 0x03, 24, 0 },
        };



        uchar cmp_pkt[4][4] = {
            { 0x0e, 0x14, 0x03, 0x01 },
            { 0x0e, 0xff, 0x01, 60 },
            { 0x0e, 0xff, 0x02, 60 },
            { 0x0e, 0xff, 0x03, 24 },
        };
        // Read colors
        uchar* colors[3] = { light->r, light->g, light->b };
        for(int clr = 0; clr < 3; clr++){
            for(int i = 0; i < 4; i++){
                if(!usbrecv(kb, data_pkt[i + clr * 4], in_pkt[i]))
                    return -1;
                // Make sure the first four bytes match
                // see comment above
                // if(memcmp(p, data_pkt[i + clr * 4], 4)){
                if (memcmp(in_pkt[i], (kb->fwversion >= 0x0205)? cmp_pkt[i] : data_pkt[i + clr * 4
  ], 4)) {
                    ckb_err("Bad input header\n");
                    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,
                        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],
                            // data_pkt[i + clr * 4][0],    data_pkt[i + clr * 4 ][1],  data_pkt[i + clr *
   4 ][2],  data_pkt[i + clr * 4 ][3]);
                        cmp_pkt[i][0], cmp_pkt[i][1], cmp_pkt[i][2], cmp_pkt[i][3]);
                    in_pkt[2][0] = 0x99;
                    in_pkt[2][1] = 0x99;
                    in_pkt[2][2] = 0x99;
                    in_pkt[2][3] = 0x99;
                    usbrecv(kb, in_pkt[2], in_pkt[2]); // just to find it in the wireshark log
                    return -1;
                }
            }
            // Copy colors to lighting. in_pkt[0] is irrelevant.
            memcpy(colors[clr], in_pkt[1] + 4, 60);
            memcpy(colors[clr] + 60, in_pkt[2] + 4, 60);
            memcpy(colors[clr] + 120, in_pkt[3] + 4, 24);
        }
    } else {
        uchar data_pkt[5][MSG_SIZE] = {
            { 0x0e, 0x14, 0x02, 0x01, 0x01, mode + 1, 0 },
            { 0xff, 0x01, 60, 0 },
            { 0xff, 0x02, 60, 0 },
            { 0xff, 0x03, 60, 0 },
            { 0xff, 0x04, 36, 0 },
        };
        uchar in_pkt[4][MSG_SIZE] = {
            { 0xff, 0x01, 60, 0 },
            { 0xff, 0x02, 60, 0 },
            { 0xff, 0x03, 60, 0 },
            { 0xff, 0x04, 36, 0 },
        };
        // Write initial packet
        if(!usbsend(kb, data_pkt[0], 1))
            return -1;
        // Read colors
        for(int i = 1; i < 5; i++){
            if(!usbrecv(kb, data_pkt[i],in_pkt[i - 1]))
                return -1;
            if(memcmp(in_pkt[i - 1], data_pkt[i], 4)){
                ckb_err("Bad input header\n");
                return -1;
            }
        }
        // Copy the data back to the mode
        uint8_t mr[N_KEYS_HW / 2], mg[N_KEYS_HW / 2], mb[
    N_KEYS_HW / 2];
        memcpy(mr,      in_pkt[0] +  4, 60);
        memcpy(mr + 60, in_pkt[1] +  4, 12);
        memcpy(mg,      in_pkt[1] + 16, 48);
        memcpy(mg + 48, in_pkt[2] +  4, 24);
        memcpy(mb,      in_pkt[2] + 28, 36);
        memcpy(mb + 36, in_pkt[3] +  4, 36);
        // Unpack LED data to 8bpc format
        for(int i = 0; i < N_KEYS_HW; i++){
            int     i_2 = i / 2;
            uint8_t r, g, b;

            // 3-bit intensities stored in alternate nybbles.
            if (i & 1) {
                r = 7 - (mr[i_2] >> 4);
                g = 7 - (mg[i_2] >> 4);
                b = 7 - (mb[i_2] >> 4);
            } else {
                r = 7 - (mr[i_2] & 0x0F);
                g = 7 - (mg[i_2] & 0x0F);
                b = 7 - (mb[i_2] & 0x0F);
            }
            // Scale 3-bit values up to 8 bits.
            light->r[i] = r << 5 | r << 2 | r >> 1;
            light->g[i] = g << 5 | g << 2 | g >> 1;
            light->b[i] = b << 5 | b << 2 | b >> 1;
        }
    }
    return 0;
}
```

Here is the caller graph for this function:



5.25.1.7 int loadrgb_mouse (usbdevice ∗ *kb,* lighting ∗ *light,* int *mode*)

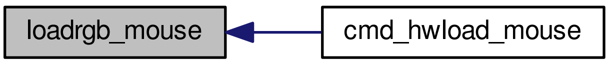
Definition at line 81 of file led_mouse.c.

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().

```
                                                                  {
    uchar data_pkt[MSG_SIZE] = { 0x0e, 0x13, 0x10, 1, 0 };
    uchar in_pkt[MSG_SIZE] = { 0 };
    // Load each RGB zone
    int zonecount = IS_SCIMITAR(kb) ? 4 : IS_SABRE(kb) ? 3 : 2;
    for(int i = 0; i < zonecount; i++){
        if(!usbrecv(kb, data_pkt, in_pkt))
            return -1;
        if(memcmp(in_pkt, data_pkt, 4)){
            ckb_err("Bad input header\n");
            return -2;
        }
        // Copy data
        int led = LED_MOUSE + i;
        if(led >= LED_DPI)
            led++;          // Skip DPI light
        light->r[led] = in_pkt[4];
        light->g[led] = in_pkt[5];
        light->b[led] = in_pkt[6];
        // Set packet for next zone
        data_pkt[2]++;
    }
    return 0;
}
```

Here is the caller graph for this function:



5.25.1.8 char∗ printrgb (const lighting ∗ *light*, const usbdevice ∗ *kb*)

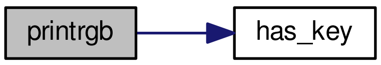
Definition at line 102 of file led.c.

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

Referenced by _cmd_get().

```
                                                          {
    uchar r[N_KEYS_EXTENDED], g[N_KEYS_EXTENDED], b[
  N_KEYS_EXTENDED];
    const uchar* mr = light->r;
    const uchar* mg = light->g;
    const uchar* mb = light->b;
    for(int i = 0; i < N_KEYS_EXTENDED; i++){
        // Translate the key index to an RGB index using the key map
        int k = keymap[i].led;
        if(k < 0)
            continue;
        r[i] = mr[k];
        g[i] = mg[k];
        b[i] = mb[k];
    }
    // Make a buffer to track key names and to filter out duplicates
    char names[N_KEYS_EXTENDED][11];
    for(int i = 0; i < N_KEYS_EXTENDED; i++){
        const char* name = keymap[i].name;
        if(keymap[i].led < 0 || !has_key(name, kb))
            names[i][0] = 0;
        else
            strncpy(names[i], name, 11);
    }
    // Check to make sure these aren't all the same color
    int same = 1;
    for(int i = 1; i < N_KEYS_EXTENDED; i++){
        if(!names[i][0])
            continue;
        if(r[i] != r[0] || g[i] != g[0] || b[i] != b[0]){
            same = 0;
            break;
        }
    }
    // If they are, just output that color
    if(same){
        char* buffer = malloc(7);
        snprintf(buffer, 7, "%02x%02x%02x", r[0], g[0], b[0]);
        return buffer;
    }
    const int BUFFER_LEN = 4096;    // Should be more than enough to fit all keys
    char* buffer = malloc(BUFFER_LEN);
    int length = 0;
    for(int i = 0; i < N_KEYS_EXTENDED; i++){
        if(!names[i][0])
            continue;
        // Print the key name
        int newlen = 0;
        snprintf(buffer + length, BUFFER_LEN - length, length == 0 ? "%s%n" : " %s%n", names[i], &newlen);
        length += newlen;
        // Look ahead to see if any other keys have this color. If so, print them here as well.
        uchar kr = r[i], kg = g[i], kb = b[i];
        for(int j = i + 1; j < N_KEYS_EXTENDED; j++){
            if(!names[j][0])
                continue;
            if(r[j] != kr || g[j] != kg || b[j] != kb)
                continue;
            snprintf(buffer + length, BUFFER_LEN - length, ",%s%n", names[j], &newlen);
            length += newlen;
            // Erase the key's name so it won't get printed later
            names[j][0] = 0;
        }
        // Print the color
        snprintf(buffer + length, BUFFER_LEN - length, ":%02x%02x%02x%n", kr, kg, kb, &newlen);
        length += newlen;
    }
    return buffer;
}
```

Here is the call graph for this function:



Here is the caller graph for this function:

printrgb
_cmd_get
cmd_get

5.25.1.9 int savergb_kb (usbdevice ∗ *kb,* lighting ∗ *light,* int *mode*)

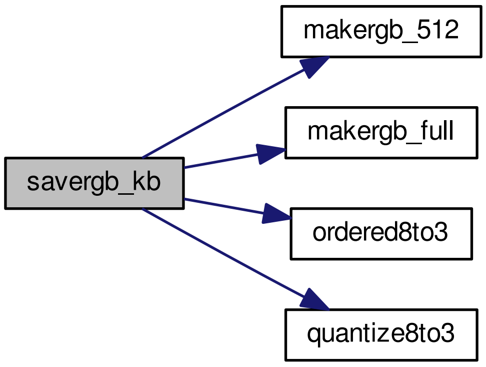
Definition at line 139 of file led_keyboard.c.

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

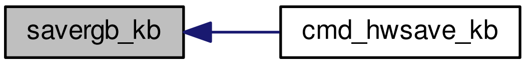
Referenced by cmd_hwsave_kb().

```
                                                        {
    if(kb->fwversion >= 0x0120){
        uchar data_pkt[12][MSG_SIZE] = {
            // Red
            { 0x7f, 0x01, 60, 0 },
            { 0x7f, 0x02, 60, 0 },
            { 0x7f, 0x03, 24, 0 },
            { 0x07, 0x14, 0x03, 0x01, 0x01, mode + 1, 0x01 },
            // Green
            { 0x7f, 0x01, 60, 0 },
            { 0x7f, 0x02, 60, 0 },
            { 0x7f, 0x03, 24, 0 },
            { 0x07, 0x14, 0x03, 0x01, 0x01, mode + 1, 0x02 },
            // Blue
            { 0x7f, 0x01, 60, 0 },
            { 0x7f, 0x02, 60, 0 },
            { 0x7f, 0x03, 24, 0 },
            { 0x07, 0x14, 0x03, 0x01, 0x01, mode + 1, 0x03 }
        };
        makergb_full(light, data_pkt);
        if(!usbsend(kb, data_pkt[0], 12))
            return -1;
        if (IS_STRAFE(kb)){ // end save
            uchar save_end_pkt[MSG_SIZE] = { 0x07, 0x14, 0x04, 0x01, 0x01 };
            if(!usbsend(kb, save_end_pkt, 1))
                return -1;
        }
    } else {
        uchar data_pkt[5][MSG_SIZE] = {
            { 0x7f, 0x01, 60, 0 },
            { 0x7f, 0x02, 60, 0 },
            { 0x7f, 0x03, 60, 0 },
            { 0x7f, 0x04, 36, 0 },
            { 0x07, 0x14, 0x02, 0x00, 0x01, mode + 1 }
        };
        makergb_512(light, data_pkt, kb->dither ? ordered8to3 :
  quantize8to3);
        if(!usbsend(kb, data_pkt[0], 5))
            return -1;
    }
    return 0;
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.25.1.10 int savergb_mouse (**usbdevice** ∗ *kb,* **lighting** ∗ *light,* int *mode*)

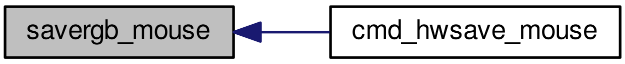
Definition at line 62 of file led_mouse.c.

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

Referenced by cmd_hwsave_mouse().

```
                                                                                      {
       uchar data_pkt[MSG_SIZE] = { 0x07, 0x13, 0x10, 1, 0 };
       // Save each RGB zone, minus the DPI light which is sent in the DPI packets
       int zonecount = IS_SCIMITAR(kb) ? 4 : IS_SABRE(kb) ? 3 : 2;
       for(int i = 0; i < zonecount; i++){
           int led = LED_MOUSE + i;
           if(led >= LED_DPI)
               led++;           // Skip DPI light
           data_pkt[4] = light->r[led];
           data_pkt[5] = light->g[led];
           data_pkt[6] = light->b[led];
           if(!usbsend(kb, data_pkt, 1))
               return -1;
           // Set packet for next zone
           data_pkt[2]++;
       }
       return 0;
   }
```

Here is the caller graph for this function:



5.25.1.11 int updatergb_kb (**usbdevice** ∗ *kb,* int *force*)

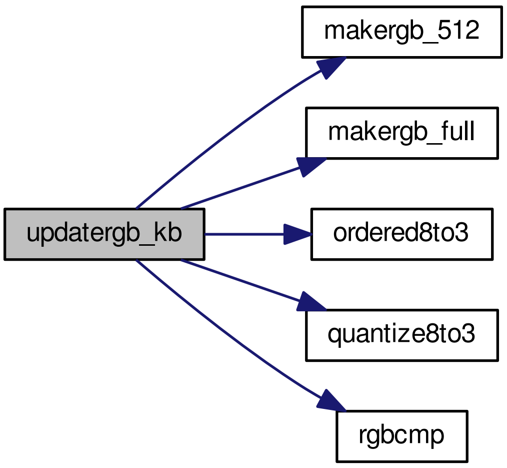
Definition at line 77 of file led_keyboard.c.

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.

```
                                                              {
```

```
    if(!kb->active)
        return 0;
    lighting* lastlight = &kb->profile->lastlight;
    lighting* newlight = &kb->profile->currentmode->
   light;
    // Don't do anything if the lighting hasn't changed
    if(!force && !lastlight->forceupdate && !newlight->forceupdate
            && !rgbcmp(lastlight, newlight) && lastlight->sidelight == newlight->
   sidelight)   // strafe sidelights
        return 0;
    lastlight->forceupdate = newlight->forceupdate = 0;

    if(IS_FULLRANGE(kb)){
        // Update strafe sidelights if necessary
        if(lastlight->sidelight != newlight->sidelight) {
            uchar data_pkt[2][MSG_SIZE] = {
                 { 0x07, 0x05, 0x08, 0x00, 0x00 },
                 { 0x07, 0x05, 0x02, 0, 0x03 }
             };
             if (newlight->sidelight)
                 data_pkt[0][4]=1;    // turn on
             if(!usbsend(kb, data_pkt[0], 2))
                 return -1;
        }
         // 16.8M color lighting works fine on strafe and is the only way it actually works
         uchar data_pkt[12][MSG_SIZE] = {
             // Red
             { 0x7f, 0x01, 0x3c, 0 },
             { 0x7f, 0x02, 0x3c, 0 },
             { 0x7f, 0x03, 0x18, 0 },
             { 0x07, 0x28, 0x01, 0x03, 0x01, 0},
             // Green
             { 0x7f, 0x01, 0x3c, 0 },
             { 0x7f, 0x02, 0x3c, 0 },
             { 0x7f, 0x03, 0x18, 0 },
             { 0x07, 0x28, 0x02, 0x03, 0x01, 0},
             // Blue
             { 0x7f, 0x01, 0x3c, 0 },
             { 0x7f, 0x02, 0x3c, 0 },
             { 0x7f, 0x03, 0x18, 0 },
             { 0x07, 0x28, 0x03, 0x03, 0x02, 0}
         };
         makergb_full(newlight, data_pkt);
         if(!usbsend(kb, data_pkt[0], 12))
             return -1;
     } else {
         // On older keyboards it looks flickery and causes lighting glitches, so we don't use it.
         uchar data_pkt[5][MSG_SIZE] = {
             { 0x7f, 0x01, 60, 0 },
             { 0x7f, 0x02, 60, 0 },
             { 0x7f, 0x03, 60, 0 },
             { 0x7f, 0x04, 36, 0 },
             { 0x07, 0x27, 0x00, 0x00, 0xD8 }
         };
         makergb_512(newlight, data_pkt, kb->dither ?
   ordered8to3 : quantize8to3);
         if(!usbsend(kb, data_pkt[0], 5))
             return -1;
     }

     memcpy(lastlight, newlight, sizeof(lighting));
     return 0;
 }
```

Here is the call graph for this function:



5.25.1.12 int updatergb_mouse (**usbdevice** ∗ *kb,* int *force*)

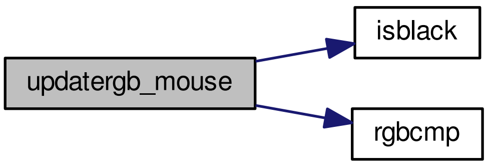
Definition at line 20 of file led_mouse.c.

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.

```
                                              {
    if(!kb->active)
        return 0;
    lighting* lastlight = &kb->profile->lastlight;
    lighting* newlight = &kb->profile->currentmode->
   light;
    // Don't do anything if the lighting hasn't changed
    if(!force && !lastlight->forceupdate && !newlight->forceupdate
            && !rgbcmp(lastlight, newlight))
        return 0;
    lastlight->forceupdate = newlight->forceupdate = 0;

    // Send the RGB values for each zone to the mouse
    uchar data_pkt[2][MSG_SIZE] = {
        { 0x07, 0x22, N_MOUSE_ZONES, 0x01, 0 }, // RGB colors
        { 0x07, 0x05, 0x02, 0 }                 // Lighting on/off
    };
    uchar* rgb_data = &data_pkt[0][4];
    for(int i = 0; i < N_MOUSE_ZONES; i++){
        *rgb_data++ = i + 1;
        *rgb_data++ = newlight->r[LED_MOUSE + i];
        *rgb_data++ = newlight->g[LED_MOUSE + i];
        *rgb_data++ = newlight->b[LED_MOUSE + i];
    }
    // Send RGB data
    if(!usbsend(kb, data_pkt[0], 1))
        return -1;
    int was_black = isblack(kb, lastlight), is_black = isblack(kb, newlight);
    if(is_black){
        // If the lighting is black, send the deactivation packet (M65 only)
        if(!usbsend(kb, data_pkt[1], 1))
            return -1;
    } else if(was_black || force){
        // If the lighting WAS black, or if we're on forced update, send the activation packet
        data_pkt[1][4] = 1;
        if(!usbsend(kb, data_pkt[1], 1))
```

```
            return -1;
    }

    memcpy(lastlight, newlight, sizeof(lighting));
    return 0;
}
```

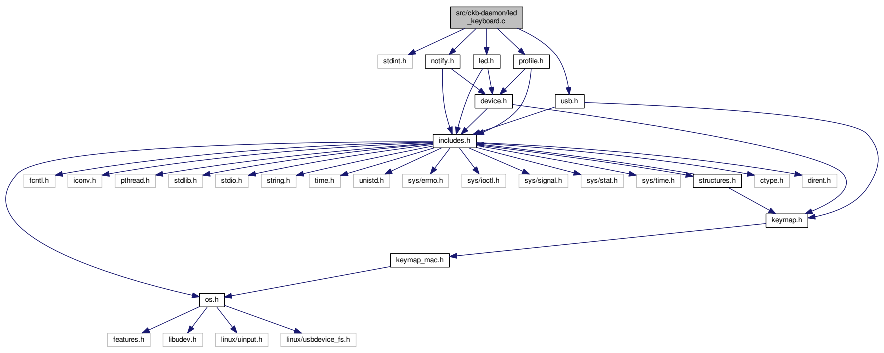
Here is the call graph for this function:



5.26 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) & 0xaa) >> 1) | ((((((0) + 32) + 8) + 4) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((0) + 32) + 8) + 4)
& 0xaa) >> 1) | ((((((0) + 32) + 8) + 4) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((0)
+ 32) + 8) + 4) & 0xaa) >> 1) | ((((((0) + 32) + 8) + 4) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((0) +
32) + 8) + 4) & 0xaa) >> 1) | ((((((0) + 32) + 8) + 4) & 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 ) & 0xaa) >> 1) | ((( ( ( ( ( ( 0 ) + 64 ) + 16 ) + 8 ) + 4 ) + 2 ) & 0x55) << 1)) & 0xcc) >> 2) | (((((( (
( ( ( ( 0 ) + 64 ) + 16 ) + 8 ) + 4 ) + 2 ) & 0xaa) >> 1) | ((( ( ( ( ( ( 0 ) + 64 ) + 16 ) + 8 ) + 4 ) + 2 ) & 0x55) <<
1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((( ( ( ( ( ( 0 ) + 64 ) + 16 ) + 8 ) + 4 ) + 2 ) & 0xaa) >> 1) | ((( ( ( ( ( (
0 ) + 64 ) + 16 ) + 8 ) + 4 ) + 2 ) & 0x55) << 1)) & 0xcc) >> 2) | (((((( ( ( ( ( ( 0 ) + 64 ) + 16 ) + 8 ) + 4 ) + 2 )
& 0xaa) >> 1) | ((( ( ( ( ( ( 0 ) + 64 ) + 16 ) + 8 ) + 4 ) + 2 ) & 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) + 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)) & 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)), }

5.26.1 Macro Definition Documentation

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

Definition at line 9 of file led_keyboard.c.

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

Definition at line 10 of file led_keyboard.c.

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

Definition at line 11 of file led_keyboard.c.

5.26.1.4 #define O0(*i*) BR4(i),

Definition at line 12 of file led_keyboard.c.

5.26.1.5 #define O1(*i*) O0(i) O0((i) + 1)

Definition at line 13 of file led_keyboard.c.

5.26.1.6 #define O2(*i*) O1(i) O1((i) + 2)

Definition at line 14 of file led_keyboard.c.

5.26.1.7 #define O3(*i*) O2(i) O2((i) + 4)

Definition at line 15 of file led_keyboard.c.

5.26.1.8 #define O4(*i*) O3(i) O3((i) + 8)

Definition at line 16 of file led_keyboard.c.

5.26.1.9 #define O5(*i*) O4(i) O4((i) + 16)

Definition at line 17 of file led_keyboard.c.

5.26.1.10 #define O6(*i*) O5(i) O5((i) + 32)

Definition at line 18 of file led_keyboard.c.

5.26.1.11 #define O7(*i*) O6(i) O6((i) + 64)

Definition at line 19 of file led_keyboard.c.

5.26.1.12 #define O8(*i*) O7(i) O7((i) + 127)

Definition at line 20 of file led_keyboard.c.

5.26.2 Function Documentation

5.26.2.1 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.

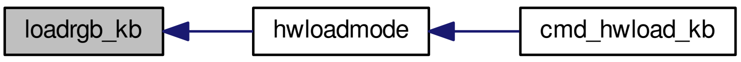
Definition at line 181 of file led_keyboard.c.

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 },
        };
        uchar in_pkt[4][MSG_SIZE] = {
            { 0xff, 0x01, 60, 0 },
            { 0xff, 0x02, 60, 0 },
            { 0xff, 0x03, 60, 0 },
            { 0xff, 0x04, 36, 0 },
        };
        // Write initial packet
        if(!usbsend(kb, data_pkt[0], 1))
            return -1;
        // Read colors
        for(int i = 1; i < 5; i++){
            if(!usbrecv(kb, data_pkt[i],in_pkt[i - 1]))
                return -1;
            if(memcmp(in_pkt[i - 1], data_pkt[i], 4)){
                ckb_err("Bad input header\n");
                return -1;
            }
        }
        // Copy the data back to the mode
        uint8_t mr[N_KEYS_HW / 2], mg[N_KEYS_HW / 2], mb[
  N_KEYS_HW / 2];
        memcpy(mr,      in_pkt[0] +  4, 60);
        memcpy(mr + 60, in_pkt[1] +  4, 12);
        memcpy(mg,      in_pkt[1] + 16, 48);
        memcpy(mg + 48, in_pkt[2] +  4, 24);
        memcpy(mb,      in_pkt[2] + 28, 36);
        memcpy(mb + 36, in_pkt[3] +  4, 36);
        // Unpack LED data to 8bpc format
        for(int i = 0; i < N_KEYS_HW; i++){
            int     i_2 = i / 2;
            uint8_t r, g, b;

            // 3-bit intensities stored in alternate nybbles.
            if (i & 1) {
                r = 7 - (mr[i_2] >> 4);
                g = 7 - (mg[i_2] >> 4);
                b = 7 - (mb[i_2] >> 4);
            } else {
                r = 7 - (mr[i_2] & 0x0F);
                g = 7 - (mg[i_2] & 0x0F);
                b = 7 - (mb[i_2] & 0x0F);
            }
            // Scale 3-bit values up to 8 bits.
            light->r[i] = r << 5 | r << 2 | r >> 1;
            light->g[i] = g << 5 | g << 2 | g >> 1;
            light->b[i] = b << 5 | b << 2 | b >> 1;
        }
    }
    return 0;
}
```

Here is the caller graph for this function:



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

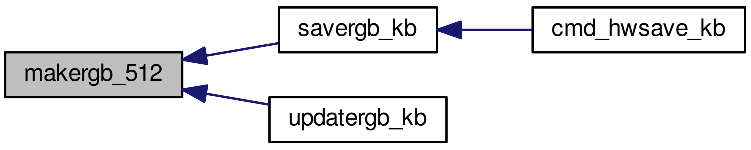
Definition at line 36 of file led_keyboard.c.

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

Referenced by savergb_kb(), and updatergb_kb().

```
                                                        {
    uchar r[N_KEYS_HW / 2], g[N_KEYS_HW / 2], b[N_KEYS_HW / 2];
    // Compress RGB values to a 512-color palette
    for(int i = 0; i < N_KEYS_HW; i += 2){
        char r1 = ditherfn(i, light->r[i]), r2 = ditherfn(i + 1, light->r[i + 1]);
        char g1 = ditherfn(i, light->g[i]), g2 = ditherfn(i + 1, light->g[i + 1]);
        char b1 = ditherfn(i, light->b[i]), b2 = ditherfn(i + 1, light->b[i + 1]);
        r[i / 2] = (7 - r2) << 4 | (7 - r1);
        g[i / 2] = (7 - g2) << 4 | (7 - g1);
        b[i / 2] = (7 - b2) << 4 | (7 - b1);
    }
    memcpy(data_pkt[0] + 4, r, 60);
    memcpy(data_pkt[1] + 4, r + 60, 12);
    memcpy(data_pkt[1] + 16, g, 48);
    memcpy(data_pkt[2] + 4, g + 48, 24);
    memcpy(data_pkt[2] + 28, b, 36);
    memcpy(data_pkt[3] + 4, b + 36, 36);
}
```

Here is the caller graph for this function:



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

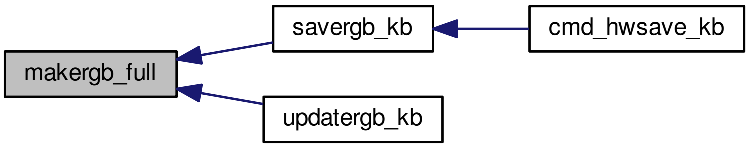
Definition at line 56 of file led_keyboard.c.

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

Referenced by savergb_kb(), and updatergb_kb().

```
                                                                         {
    const uchar* r = light->r, *g = light->g, *b = light->b;
    // Red
    memcpy(data_pkt[0] + 4, r, 60);
    memcpy(data_pkt[1] + 4, r + 60, 60);
    memcpy(data_pkt[2] + 4, r + 120, 24);
    // Green (final R packet is blank)
    memcpy(data_pkt[4] + 4, g, 60);
    memcpy(data_pkt[5] + 4, g + 60, 60);
    memcpy(data_pkt[6] + 4, g + 120, 24);
    // Blue (final G packet is blank)
    memcpy(data_pkt[8] + 4, b, 60);
    memcpy(data_pkt[9] + 4, b + 60, 60);
    memcpy(data_pkt[10] + 4, b + 120, 24);
}
```

Here is the caller graph for this function:



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

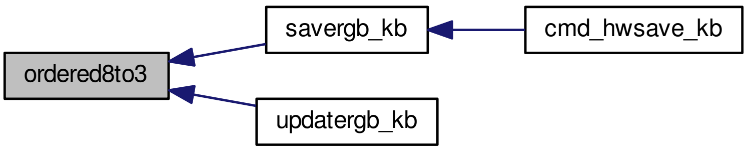
Definition at line 24 of file led_keyboard.c.

References bit_reverse_table.

Referenced by savergb_kb(), and updatergb_kb().

```
                                                {
    int m = value * 7;
    int b = m / 255;
    if((m % 255) > bit_reverse_table[index & 0xff])
        b++;
    return b;
}
```

Here is the caller graph for this function:



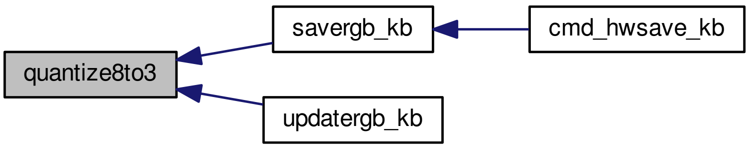
5.26.2.5 static uchar quantize8to3 (int *index*, uchar *value*) [static]

Definition at line 32 of file led_keyboard.c.

Referenced by savergb_kb(), and updatergb_kb().

```
                                                {
    return value >> 5;
}
```

Here is the caller graph for this function:



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

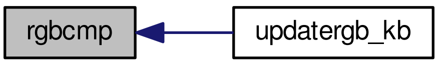
Definition at line 72 of file led_keyboard.c.

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

Referenced by updatergb_kb().

```
                                                              {
      // Compare two light structures, ignore mouse zones
      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);
 }
```

Here is the caller graph for this function:



5.26.2.7 int savergb_kb (usbdevice ∗ *kb,* lighting ∗ *light,* int *mode*)

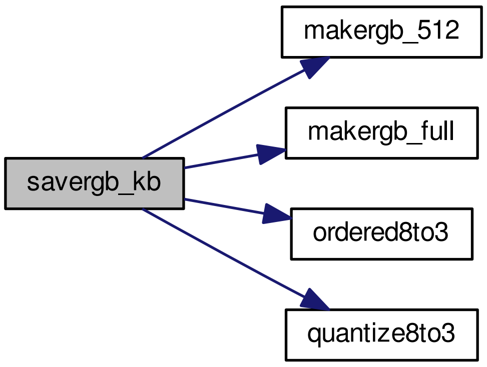
Definition at line 139 of file led_keyboard.c.

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

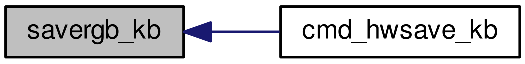
Referenced by cmd_hwsave_kb().

```
                                                         {
    if(kb->fwversion >= 0x0120){
        uchar data_pkt[12][MSG_SIZE] = {
            // Red
            { 0x7f, 0x01, 60, 0 },
            { 0x7f, 0x02, 60, 0 },
            { 0x7f, 0x03, 24, 0 },
            { 0x07, 0x14, 0x03, 0x01, 0x01, mode + 1, 0x01 },
            // Green
            { 0x7f, 0x01, 60, 0 },
            { 0x7f, 0x02, 60, 0 },
            { 0x7f, 0x03, 24, 0 },
            { 0x07, 0x14, 0x03, 0x01, 0x01, mode + 1, 0x02 },
            // Blue
            { 0x7f, 0x01, 60, 0 },
            { 0x7f, 0x02, 60, 0 },
            { 0x7f, 0x03, 24, 0 },
            { 0x07, 0x14, 0x03, 0x01, 0x01, mode + 1, 0x03 }
        };
        makergb_full(light, data_pkt);
        if(!usbsend(kb, data_pkt[0], 12))
            return -1;
        if (IS_STRAFE(kb)){ // end save
            uchar save_end_pkt[MSG_SIZE] = { 0x07, 0x14, 0x04, 0x01, 0x01 };
            if(!usbsend(kb, save_end_pkt, 1))
                return -1;
        }
    } else {
        uchar data_pkt[5][MSG_SIZE] = {
            { 0x7f, 0x01, 60, 0 },
            { 0x7f, 0x02, 60, 0 },
            { 0x7f, 0x03, 60, 0 },
            { 0x7f, 0x04, 36, 0 },
            { 0x07, 0x14, 0x02, 0x00, 0x01, mode + 1 }
        };
        makergb_512(light, data_pkt, kb->dither ? ordered8to3 :
     quantize8to3);
        if(!usbsend(kb, data_pkt[0], 5))
            return -1;
    }
    return 0;
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.26.2.8 int updatergb_kb (usbdevice ∗ *kb,* int *force*)

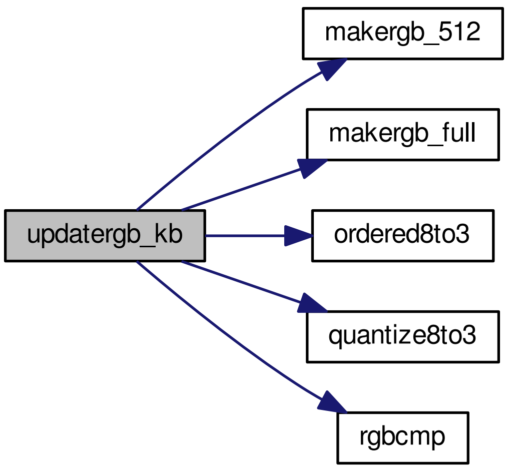
Definition at line 77 of file led_keyboard.c.

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.

```
                                        {
    if(!kb->active)
        return 0;
    lighting* lastlight = &kb->profile->lastlight;
    lighting* newlight = &kb->profile->currentmode->
   light;
    // Don't do anything if the lighting hasn't changed
    if(!force && !lastlight->forceupdate && !newlight->forceupdate
            && !rgbcmp(lastlight, newlight) && lastlight->sidelight == newlight->
   sidelight)   // strafe sidelights
        return 0;
    lastlight->forceupdate = newlight->forceupdate = 0;

    if(IS_FULLRANGE(kb)){
        // Update strafe sidelights if necessary
        if(lastlight->sidelight != newlight->sidelight) {
            uchar data_pkt[2][MSG_SIZE] = {
                 { 0x07, 0x05, 0x08, 0x00, 0x00 },
                 { 0x07, 0x05, 0x02, 0, 0x03 }
             };
             if (newlight->sidelight)
```

```
                data_pkt[0][4]=1;    // turn on
            if(!usbsend(kb, data_pkt[0], 2))
                return -1;
       }
        // 16.8M color lighting works fine on strafe and is the only way it actually works
        uchar data_pkt[12][MSG_SIZE] = {
            // Red
            { 0x7f, 0x01, 0x3c, 0 },
            { 0x7f, 0x02, 0x3c, 0 },
            { 0x7f, 0x03, 0x18, 0 },
            { 0x07, 0x28, 0x01, 0x03, 0x01, 0},
            // Green
            { 0x7f, 0x01, 0x3c, 0 },
            { 0x7f, 0x02, 0x3c, 0 },
            { 0x7f, 0x03, 0x18, 0 },
            { 0x07, 0x28, 0x02, 0x03, 0x01, 0},
            // Blue
            { 0x7f, 0x01, 0x3c, 0 },
            { 0x7f, 0x02, 0x3c, 0 },
            { 0x7f, 0x03, 0x18, 0 },
            { 0x07, 0x28, 0x03, 0x03, 0x02, 0}
        };
        makergb_full(newlight, data_pkt);
        if(!usbsend(kb, data_pkt[0], 12))
            return -1;
    } else {
        // On older keyboards it looks flickery and causes lighting glitches, so we don’t use it.
        uchar data_pkt[5][MSG_SIZE] = {
            { 0x7f, 0x01, 60, 0 },
            { 0x7f, 0x02, 60, 0 },
            { 0x7f, 0x03, 60, 0 },
            { 0x7f, 0x04, 36, 0 },
            { 0x07, 0x27, 0x00, 0x00, 0xD8 }
        };
        makergb_512(newlight, data_pkt, kb->dither ?
      ordered8to3 : quantize8to3);
        if(!usbsend(kb, data_pkt[0], 5))
            return -1;
    }

    memcpy(lastlight, newlight, sizeof(lighting));
    return 0;
}
```

Here is the call graph for this function:



5.26.3 Variable Documentation

5.26.3.1 uchar bit_reverse_table[256]

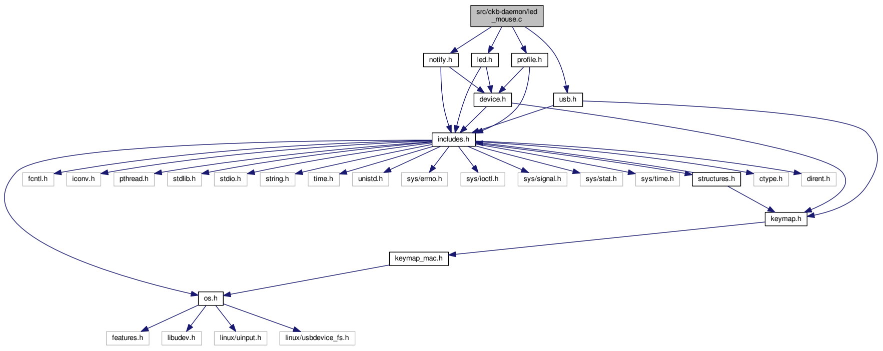
```
uchar bit_reverse_table[256] = { (((((((((( 0 ) & 0xaa) >> 1) | ((( 0 ) & 0x55) << 1)) & 0xcc) >> 2) | (((((( 0 ) & 0xaa) >>
1) | ((( 0 ) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((( 0 ) & 0xaa) >> 1) | ((( 0 ) & 0x55) << 1)) & 0xcc) >>
2) | (((((( 0 ) & 0xaa) >> 1) | ((( 0 ) & 0x55) << 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().

5.27 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)

5.27.1 Function Documentation

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

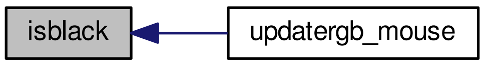
Definition at line 13 of file led_mouse.c.

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

Referenced by updatergb_mouse().

```
{
    if(!IS_M65(kb))
        return 0;
    uchar black[N_MOUSE_ZONES] = { 0 };
    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));
}
```

Here is the caller graph for this function:



5.27.1.2 int loadrgb_mouse (**usbdevice** ∗ *kb,* **lighting** ∗ *light,* int *mode*)

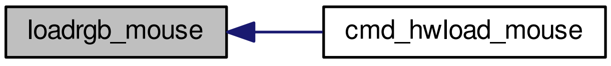
Definition at line 81 of file led_mouse.c.

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().

```
                                                           {
    uchar data_pkt[MSG_SIZE] = { 0x0e, 0x13, 0x10, 1, 0 };
    uchar in_pkt[MSG_SIZE] = { 0 };
    // Load each RGB zone
    int zonecount = IS_SCIMITAR(kb) ? 4 : IS_SABRE(kb) ? 3 : 2;
    for(int i = 0; i < zonecount; i++){
        if(!usbrecv(kb, data_pkt, in_pkt))
            return -1;
        if(memcmp(in_pkt, data_pkt, 4)){
            ckb_err("Bad input header\n");
            return -2;
        }
        // Copy data
        int led = LED_MOUSE + i;
        if(led >= LED_DPI)
            led++;          // Skip DPI light
        light->r[led] = in_pkt[4];
        light->g[led] = in_pkt[5];
        light->b[led] = in_pkt[6];
        // Set packet for next zone
        data_pkt[2]++;
    }
    return 0;
}
```

Here is the caller graph for this function:



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

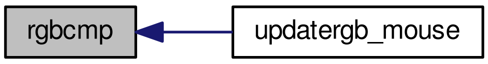
Definition at line 7 of file led_mouse.c.

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

Referenced by updatergb_mouse().

```
                                                            {
    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);
}
```

Here is the caller graph for this function:



5.27.1.4 int savergb_mouse (usbdevice ∗ *kb*, lighting ∗ *light*, int *mode*)

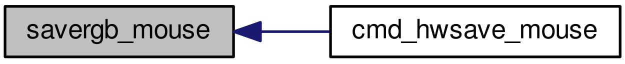
Definition at line 62 of file led_mouse.c.

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

Referenced by cmd_hwsave_mouse().

```
                                                             {
    uchar data_pkt[MSG_SIZE] = { 0x07, 0x13, 0x10, 1, 0 };
    // Save each RGB zone, minus the DPI light which is sent in the DPI packets
    int zonecount = IS_SCIMITAR(kb) ? 4 : IS_SABRE(kb) ? 3 : 2;
    for(int i = 0; i < zonecount; i++){
        int led = LED_MOUSE + i;
        if(led >= LED_DPI)
            led++;          // Skip DPI light
        data_pkt[4] = light->r[led];
        data_pkt[5] = light->g[led];
        data_pkt[6] = light->b[led];
        if(!usbsend(kb, data_pkt, 1))
            return -1;
        // Set packet for next zone
        data_pkt[2]++;
    }
    return 0;
}
```

Here is the caller graph for this function:



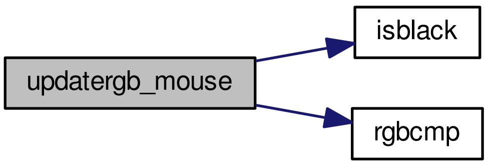
5.27.1.5 int updatergb_mouse (usbdevice ∗ *kb*, int *force*)

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.

```
                                              {
    if(!kb->active)
        return 0;
    lighting* lastlight = &kb->profile->lastlight;
    lighting* newlight = &kb->profile->currentmode->light;
    // Don't do anything if the lighting hasn't changed
    if(!force && !lastlight->forceupdate && !newlight->forceupdate
            && !rgbcmp(lastlight, newlight))
        return 0;
    lastlight->forceupdate = newlight->forceupdate = 0;

    // Send the RGB values for each zone to the mouse
    uchar data_pkt[2][MSG_SIZE] = {
        { 0x07, 0x22, N_MOUSE_ZONES, 0x01, 0 }, // RGB colors
        { 0x07, 0x05, 0x02, 0 }                 // Lighting on/off
    };
    uchar* rgb_data = &data_pkt[0][4];
    for(int i = 0; i < N_MOUSE_ZONES; i++){
        *rgb_data++ = i + 1;
        *rgb_data++ = newlight->r[LED_MOUSE + i];
        *rgb_data++ = newlight->g[LED_MOUSE + i];
        *rgb_data++ = newlight->b[LED_MOUSE + i];
    }
    // Send RGB data
    if(!usbsend(kb, data_pkt[0], 1))
        return -1;
    int was_black = isblack(kb, lastlight), is_black = isblack(kb, newlight);
    if(is_black){
        // If the lighting is black, send the deactivation packet (M65 only)
        if(!usbsend(kb, data_pkt[1], 1))
            return -1;
    } else if(was_black || force){
        // If the lighting WAS black, or if we're on forced update, send the activation packet
        data_pkt[1][4] = 1;
        if(!usbsend(kb, data_pkt[1], 1))
            return -1;
    }

    memcpy(lastlight, newlight, sizeof(lighting));
    return 0;
}
```

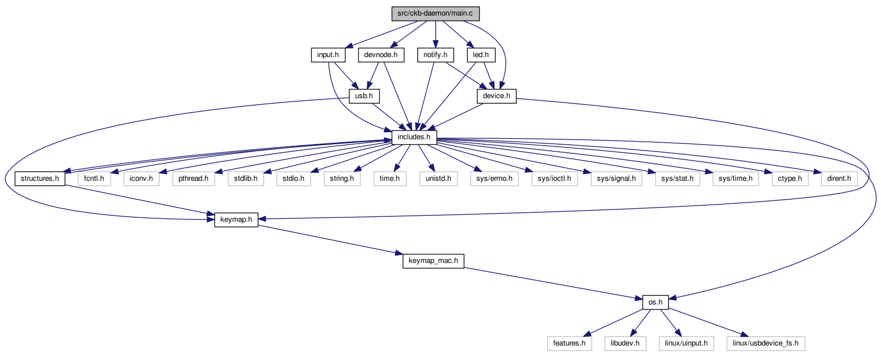
Here is the call graph for this function:



5.28 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

  brief .
- int features_mask

  brief .
- int hwload_mode

5.28.1 Function Documentation

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

Definition at line 71 of file main.c.

```
                                                                 {
    char* ldst = dst + length;
    char s;
    while((s = *src++)){
        if(s == '_')
            s = '-';
        else
            s = tolower(s);
        *dst++ = s;
        if(dst == ldst){
            dst--;
```

```
82              break;
83          }
84      }
85      *dst = 0;
86 }
```

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

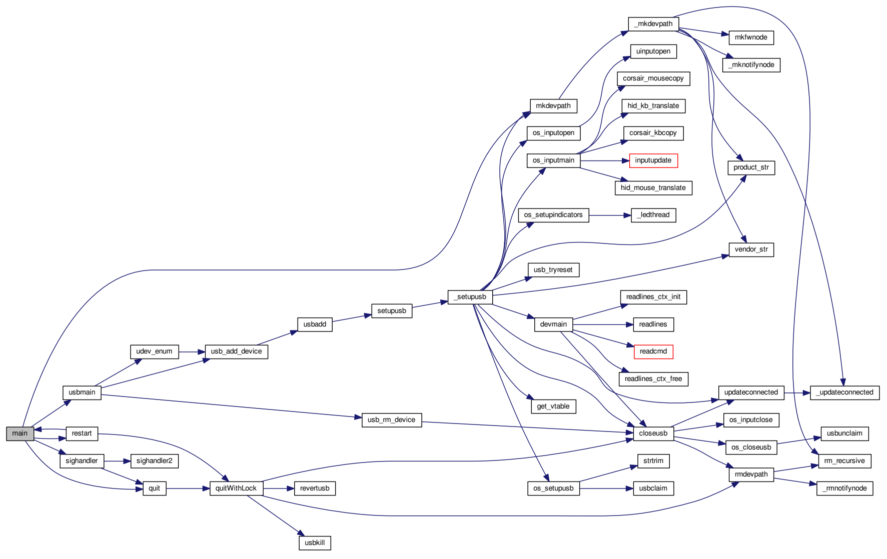
Definition at line 88 of file main.c.

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().

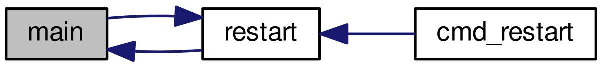
Referenced by restart().

```
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);
                return 0;
            }
        }
    }

    // Read parameters
    int forceroot = 1;
    for(int i = 1; i < argc; i++){
        char* argument = argv[i];
        unsigned newgid;
        char hwload[7];
        if(sscanf(argument, "--gid=%u", &newgid) == 1){
            // Set dev node GID
            gid = newgid;
            ckb_info_nofile("Setting /dev node gid: %u\n", newgid);
        } else if(!strcmp(argument, "--nobind")){
            // Disable key notifications and rebinding
            features_mask &= ~FEAT_BIND & ~FEAT_NOTIFY;
            ckb_info_nofile("Key binding and key notifications are disabled\n");
        } else if(!strcmp(argument, "--nonotify")){
            // Disable key notifications
            features_mask &= ~FEAT_NOTIFY;
            ckb_info_nofile("Key notifications are disabled\n");
        } else if(sscanf(argument, "--hwload=%6s", hwload) == 1){
            if(!strcmp(hwload, "always") || !strcmp(hwload, "yes") || !strcmp(hwload, "y") || !strcmp(
hwload, "a")){
                hwload_mode = 2;
                ckb_info_nofile("Setting hardware load: always\n");
            } else if(!strcmp(hwload, "tryonce") || !strcmp(hwload, "try") || !strcmp(hwload, "once") || !
strcmp(hwload, "t") || !strcmp(hwload, "o")){
                hwload_mode = 1;
                ckb_info_nofile("Setting hardware load: tryonce\n");
            } else if(!strcmp(hwload, "never") || !strcmp(hwload, "none") || !strcmp(hwload, "no") || !
strcmp(hwload, "n")){
                hwload_mode = 0;
                ckb_info_nofile("Setting hardware load: never\n");
            }
        } else if(!strcmp(argument, "--nonroot")){
            // Allow running as a non-root user
            forceroot = 0;
        }
#ifdef OS_MAC
        else if(!strcmp(argument, "--nomouseaccel")){
            // On OSX, provide an option to disable mouse acceleration
            features_mask &= ~FEAT_MOUSEACCEL;
            ckb_info_nofile("Mouse acceleration disabled\n");
        }
#endif
    }

    // Check UID
    if(getuid() != 0){
        if(forceroot){
            ckb_fatal_nofile("ckb-daemon must be run as root. Try `sudo %s`\n", argv[0]);
            exit(0);
        } else
            ckb_warn_nofile("Warning: not running as root, allowing anyway per command-line
 parameter...\n");
    }

    // Make root keyboard
    umask(0);
    memset(keyboard, 0, sizeof(keyboard));
    if(!mkdevpath(keyboard))
        ckb_info("Root controller ready at %s0\n", devpath);

    // Set signals
    sigset_t signals;
    sigfillset(&signals);
    sigdelset(&signals, SIGTERM);
    sigdelset(&signals, SIGINT);
    sigdelset(&signals, SIGQUIT);
    sigdelset(&signals, SIGUSR1);
    // Set up signal handlers for quitting the service.
    sigprocmask(SIG_SETMASK, &signals, 0);
    signal(SIGTERM, sighandler);
    signal(SIGINT, sighandler);
    signal(SIGQUIT, sighandler);
    signal(SIGUSR1, (void (*)())restart);

    // Start the USB system
    int result = usbmain();
    quit();
    return result;
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.28.1.3 static void quit ( ) [static]

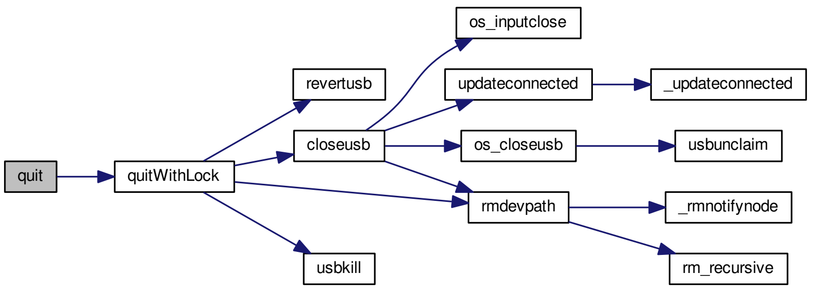
Definition at line 30 of file main.c.

References quitWithLock().

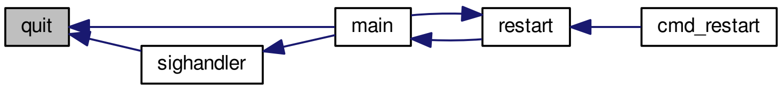
Referenced by main(), and sighandler().

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

Here is the call graph for this function:



Here is the caller graph for this function:



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

Parameters

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

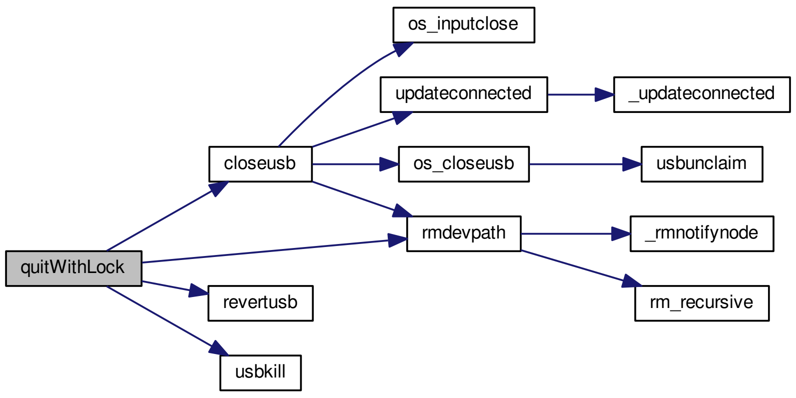
Definition at line 40 of file main.c.

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

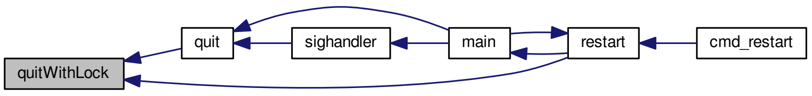
Referenced by quit(), and restart().

```
                              {
    // Abort any USB resets in progress
    reset_stop = 1;
    for(int i = 1; i < DEV_MAX; i++){
        // Before closing, set all keyboards back to HID input mode so that the stock driver can still talk
    to them
        if (mut) pthread_mutex_lock(devmutex + i);
        if(IS_CONNECTED(keyboard + i)){
            revertusb(keyboard + i);
            closeusb(keyboard + i);
        }
        pthread_mutex_unlock(devmutex + i);
    }
    ckb_info("Closing root controller\n");
    rmdevpath(keyboard);
    usbkill();
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.28.1.5 int restart ( )

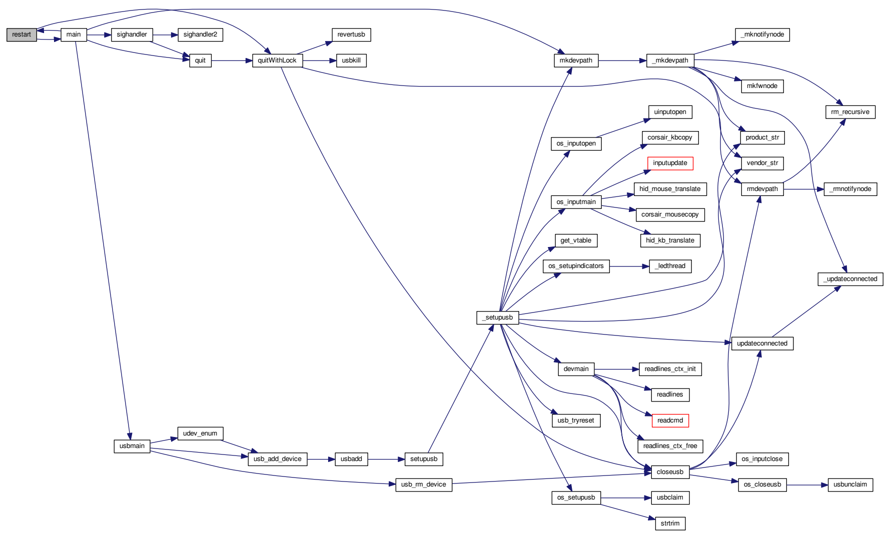
Definition at line 228 of file main.c.

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

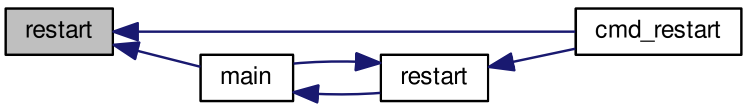
Referenced by cmd_restart(), and main().

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

Here is the call graph for this function:



Here is the caller graph for this function:



5.28.1.6 void sighandler (int *type*)

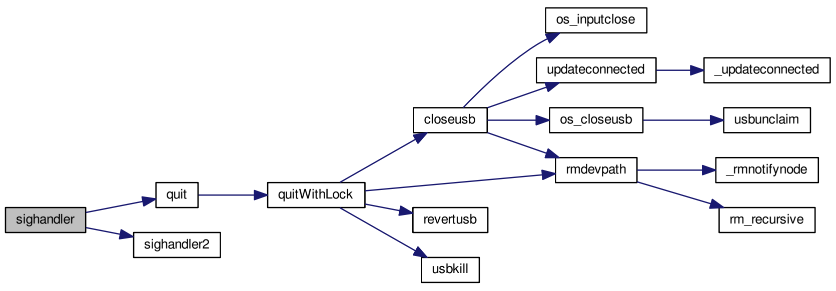
Definition at line 62 of file main.c.

References quit(), and sighandler2().

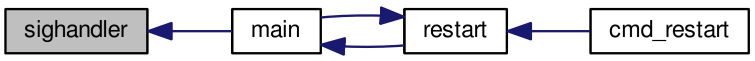
Referenced by main().

```
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:



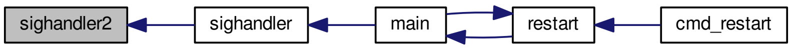
5.28.1.7 void sighandler2 (int *type*)

Definition at line 57 of file main.c.

Referenced by sighandler().

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

Here is the caller graph for this function:



5.28.1.8 void timespec_add (struct timespec ∗ *timespec,* long *nanoseconds*)

Definition at line 19 of file main.c.

```
                                                            {
    nanoseconds += timespec->tv_nsec;
    timespec->tv_sec += nanoseconds / 1000000000;
    timespec->tv_nsec = nanoseconds % 1000000000;
}
```

5.28.2 Variable Documentation

5.28.2.1 int features_mask

features_mask Mask of features to exclude from all devices

That bit mask ist set to enable all (-1). When interpreting the input parameters, some of these bits can be cleared.

At the moment binding, notifying and mouse-acceleration can be disabled via command line.

Have a look at *main()* in main.c for details.

Definition at line 35 of file usb.c.

Referenced by _setupusb(), and main().

5.28.2.2 int hwload_mode

Definition at line 7 of file device.c.

Referenced by main().

5.28.2.3 int main_ac [static]

Definition at line 7 of file main.c.

Referenced by main(), and restart().

5.28.2.4 char** main_av [static]

Definition at line 8 of file main.c.

Referenced by main(), and restart().

5.28.2.5 volatile int reset_stop

reset_stop is boolean: Reset stopper for when the program shuts down.

Is set only by *quit()* to true (1) to inform several usb_* functions to end their loops and tries.

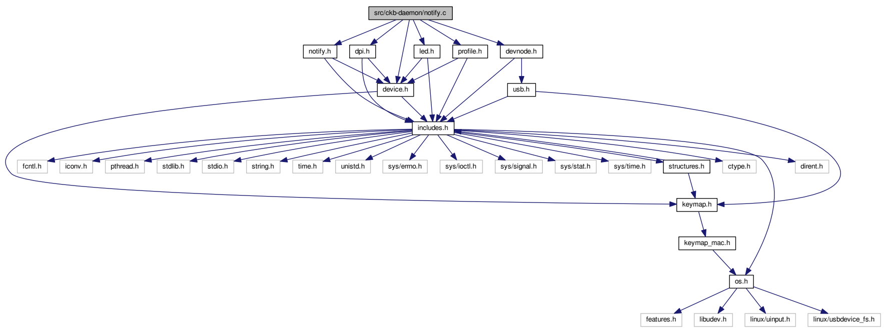
Definition at line 25 of file usb.c.

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

5.29 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)

5.29.1 Macro Definition Documentation

5.29.1.1 #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().

5.29.1.2 #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.

5.29.2 Function Documentation

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

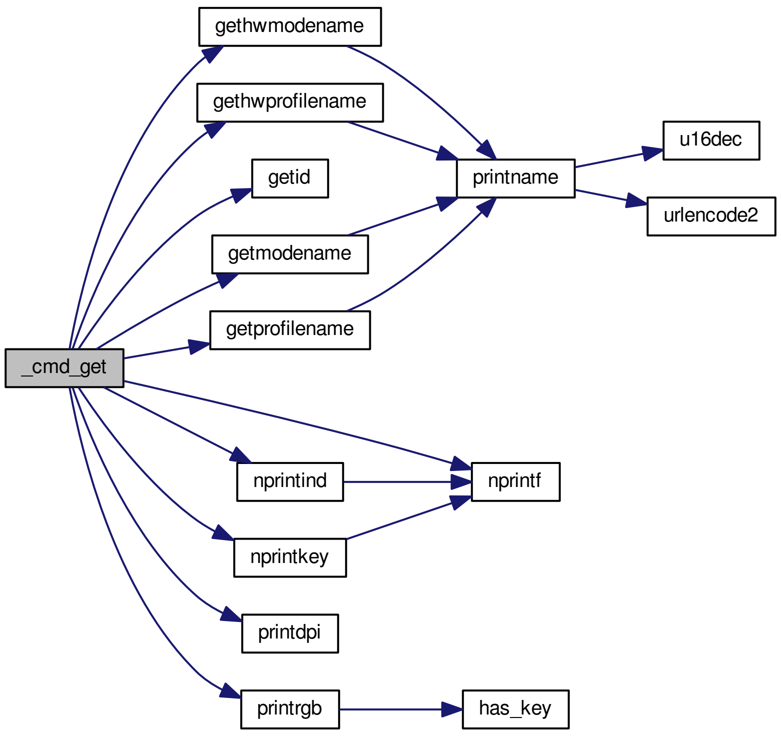
Definition at line 90 of file notify.c.

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.

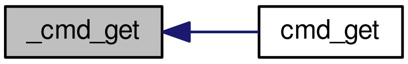
Referenced by cmd_get().

```
                                                                                  {
    usbprofile* profile = kb->profile;
    if(!strcmp(setting, ":mode")){
        // Get the current mode number
        nprintf(kb, nnumber, mode, "switch\n");
        return;
    } else if(!strcmp(setting, ":rgb")){
        // Get the current RGB settings
        char* rgb = printrgb(&mode->light, kb);
        nprintf(kb, nnumber, mode, "rgb %s\n", rgb);
         free(rgb);
         return;
     } else if(!strcmp(setting, ":hwrgb")){
         // Get the current hardware RGB settings
         HW_STANDARD;
         char* rgb = printrgb(kb->hw->light + index, kb);
         nprintf(kb, nnumber, mode, "hwrgb %s\n", rgb);
         free(rgb);
         return;
     } else if(!strcmp(setting, ":profilename")){
         // Get the current profile name
         char* name = getprofilename(profile);
         nprintf(kb, nnumber, 0, "profilename %s\n", name[0] ? name : "Unnamed");
         free(name);
     } else if(!strcmp(setting, ":name")){
         // Get the current mode name
         char* name = getmodename(mode);
         nprintf(kb, nnumber, mode, "name %s\n", name[0] ? name : "Unnamed");
         free(name);
     } else if(!strcmp(setting, ":hwprofilename")){
         // Get the current hardware profile name
         if(!kb->hw)
             return;
         char* name = gethwprofilename(kb->hw);
         nprintf(kb, nnumber, 0, "hwprofilename %s\n", name[0] ? name : "Unnamed");
         free(name);
     } else if(!strcmp(setting, ":hwname")){
         // Get the current hardware mode name
         HW_STANDARD;
         char* name = gethwmodename(kb->hw, index);
         nprintf(kb, nnumber, mode, "hwname %s\n", name[0] ? name : "Unnamed");
         free(name);
     } else if(!strcmp(setting, ":profileid")){
         // Get the current profile ID
         char* guid = getid(&profile->id);
         int modified;
         memcpy(&modified, &profile->id.modified, sizeof(modified));
         nprintf(kb, nnumber, 0, "profileid %s %x\n", guid, modified);
         free(guid);
     } else if(!strcmp(setting, ":id")){
         // Get the current mode ID
         char* guid = getid(&mode->id);
         int modified;
         memcpy(&modified, &mode->id.modified, sizeof(modified));
         nprintf(kb, nnumber, mode, "id %s %x\n", guid, modified);
         free(guid);
     } else if(!strcmp(setting, ":hwprofileid")){
         // Get the current hardware profile ID
         if(!kb->hw)
             return;
         char* guid = getid(&kb->hw->id[0]);
         int modified;
         memcpy(&modified, &kb->hw->id[0].modified, sizeof(modified));
         nprintf(kb, nnumber, 0, "hwprofileid %s %x\n", guid, modified);
         free(guid);
     } else if(!strcmp(setting, ":hwid")){
         // Get the current hardware mode ID
         HW_STANDARD;
         char* guid = getid(&kb->hw->id[index + 1]);
         int modified;
         memcpy(&modified, &kb->hw->id[index + 1].modified, sizeof(modified));
         nprintf(kb, nnumber, mode, "hwid %s %x\n", guid, modified);
         free(guid);
     } else if(!strcmp(setting, ":keys")){
         // Get the current state of all keys
         for(int i = 0; i < N_KEYS_INPUT; i++){
             if(!keymap[i].name)
                 continue;
             int byte = i / 8, bit = 1 << (i & 7);
             uchar state = kb->input.keys[byte] & bit;
             if(state)
                 nprintkey(kb, nnumber, i, 1);
         }
     } else if(!strcmp(setting, ":i")){
         // Get the current state of all indicator LEDs
         if(kb->hw_ileds & I_NUM) nprintind(kb, nnumber, I_NUM, 1);
         if(kb->hw_ileds & I_CAPS) nprintind(kb, nnumber, I_CAPS, 1);
         if(kb->hw_ileds & I_SCROLL) nprintind(kb, nnumber, I_SCROLL, 1);
     } else if(!strcmp(setting, ":dpi")){
         // Get the current DPI levels
         char* dpi = printdpi(&mode->dpi, kb);
         nprintf(kb, nnumber, mode, "dpi %s\n", dpi);
         free(dpi);
         return;
     } else if(!strcmp(setting, ":hwdpi")){
         // Get the current hardware DPI levels
         HW_STANDARD;
         char* dpi = printdpi(kb->hw->dpi + index, kb);
         nprintf(kb, nnumber, mode, "hwdpi %s\n", dpi);
         free(dpi);
         return;
     } else if(!strcmp(setting, ":dpisel")){
         // Get the currently-selected DPI
         nprintf(kb, nnumber, mode, "dpisel %d\n", mode->dpi.current);
     } else if(!strcmp(setting, ":hwdpisel")){
         // Get the currently-selected hardware DPI
         HW_STANDARD;
         nprintf(kb, nnumber, mode, "hwdpisel %d\n", kb->hw->dpi[index].current);
     } else if(!strcmp(setting, ":lift")){
         // Get the mouse lift height
         nprintf(kb, nnumber, mode, "lift %d\n", mode->dpi.lift);
     } else if(!strcmp(setting, ":hwlift")){
         // Get the hardware lift height
         HW_STANDARD;
         nprintf(kb, nnumber, mode, "hwlift %d\n", kb->hw->dpi[index].lift);
     } else if(!strcmp(setting, ":snap")){
         // Get the angle snap status
         nprintf(kb, nnumber, mode, "snap %s\n", mode->dpi.snap ? "on" : "off");
     } else if(!strcmp(setting, ":hwsnap")){
         // Get the hardware angle snap status
         HW_STANDARD;
         nprintf(kb, nnumber, mode, "hwsnap %s\n", kb->hw->dpi[index].snap ? "on" : "off");
     }
 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



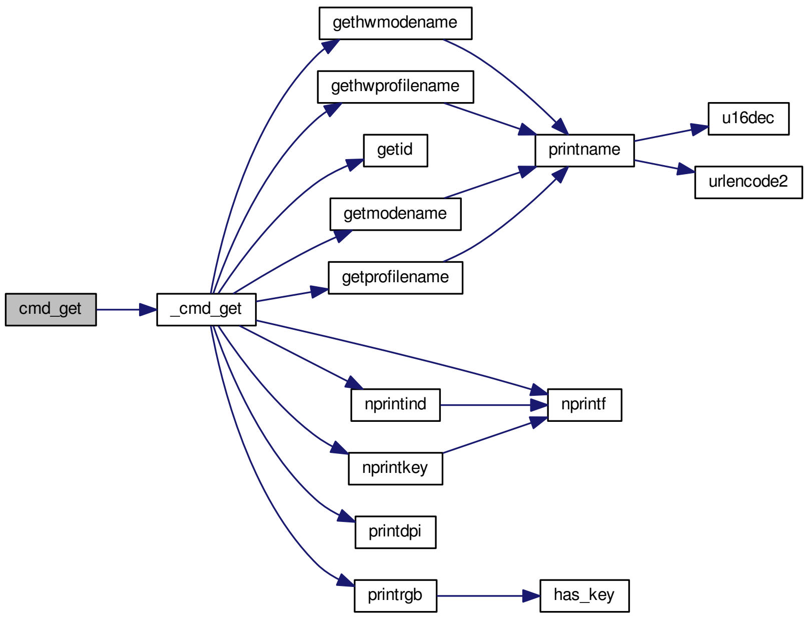
5.29.2.2 void cmd_get (usbdevice ∗ *kb*, usbmode ∗ *mode*, int *nnumber*, int *dummy*, const char ∗ *setting*)

Definition at line 215 of file notify.c.

References _cmd_get(), and imutex.

```
                                                                                         {
    pthread_mutex_lock(imutex(kb));
    _cmd_get(kb, mode, nnumber, setting);
    pthread_mutex_unlock(imutex(kb));
}
```

Here is the call graph for this function:



5.29.2.3 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.

```
                                                                                  {
    if(keyindex >= N_KEYS_INPUT)
        return;
    pthread_mutex_lock(imutex(kb));
    if(!strcmp(toggle, "on") || *toggle == 0)
        SET_KEYBIT(mode->notify[nnumber], keyindex);
    else if(!strcmp(toggle, "off"))
        CLEAR_KEYBIT(mode->notify[nnumber], keyindex);
    pthread_mutex_unlock(imutex(kb));
}
```

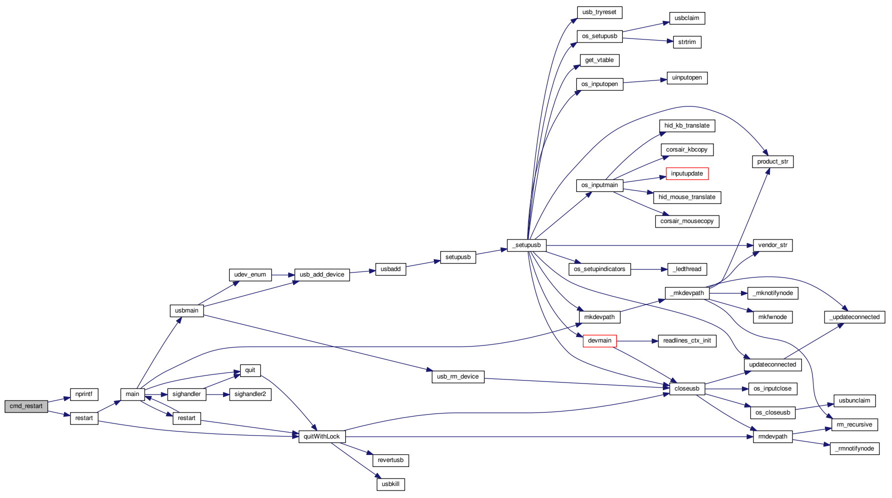
5.29.2.4 void cmd_restart (usbdevice ∗ *kb*, usbmode ∗ *mode*, int *nnumber*, int *dummy*, const char ∗ *content*)

Definition at line 223 of file notify.c.

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

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

Here is the call graph for this function:



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

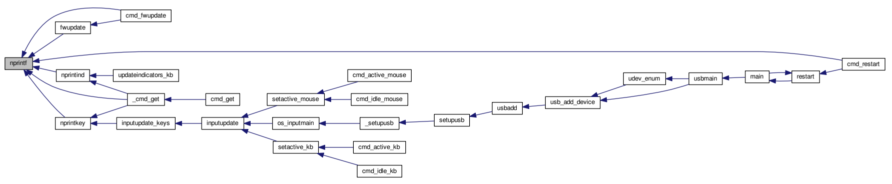
Definition at line 8 of file notify.c.

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

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

```
                                                                                    {
    if(!kb)
        return;
     usbprofile* profile = kb->profile;
     va_list va_args;
     int fifo;
     if(nodenumber >= 0){
         // If node number was given, print to that node (if open)
         if((fifo = kb->outfifo[nodenumber] - 1) != -1){
             va_start(va_args, format);
             if(mode)
                 dprintf(fifo, "mode %d ", INDEX_OF(mode, profile->mode) + 1);
             vdprintf(fifo, format, va_args);
         }
         return;
     }
     // Otherwise, print to all nodes
     for(int i = 0; i < OUTFIFO_MAX; i++){
         if((fifo = kb->outfifo[i] - 1) != -1){
             va_start(va_args, format);
             if(mode)
                 dprintf(fifo, "mode %d ", INDEX_OF(mode, profile->mode) + 1);
             vdprintf(fifo, format, va_args);
         }
     }
 }
```

Here is the caller graph for this function:



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

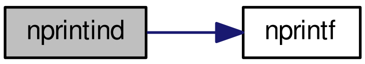
Definition at line 43 of file notify.c.

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

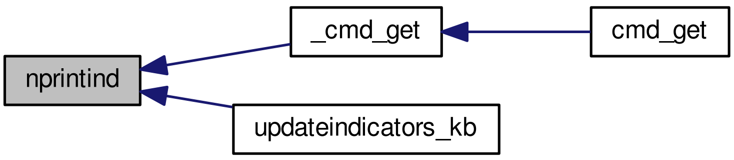
Referenced by _cmd_get(), and updateindicators_kb().

```
                                                              {
    const char* name = 0;
    switch(led){
    case I_NUM:
        name = "num";
        break;
    case I_CAPS:
        name = "caps";
        break;
    case I_SCROLL:
        name = "scroll";
        break;
    default:
        return;
    }
    nprintf(kb, nnumber, 0, "i %c%s\n", on ? '+' : '-', name);
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



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

Definition at line 35 of file notify.c.

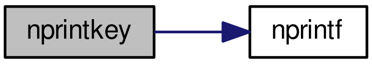
References keymap, key::name, and nprintf().

Referenced by _cmd_get(), and inputupdate_keys().

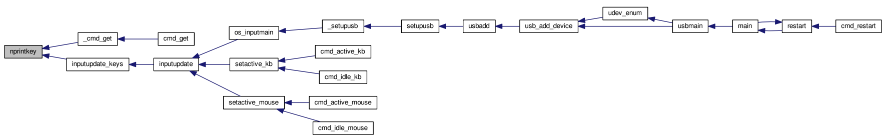
```
                                                                     {
    const key* map = keymap + keyindex;
    if(map->name)
```

```
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:



5.29.2.8 int restart ( )

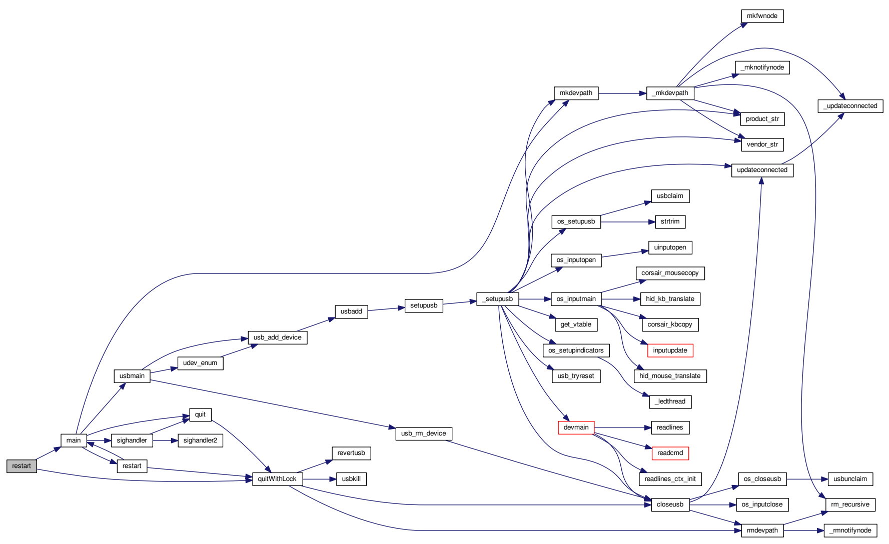
Definition at line 228 of file main.c.

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

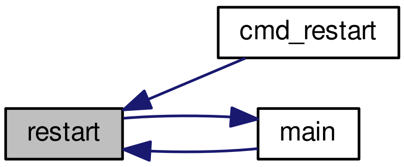
Referenced by cmd_restart(), and main().

```
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:



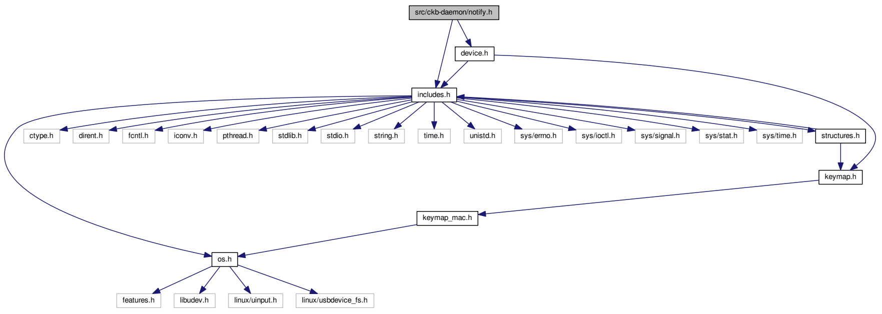
Here is the caller graph for this function:



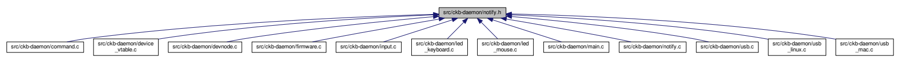
5.30 src/ckb-daemon/notify.h File Reference

```
#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)

5.30.1 Function Documentation

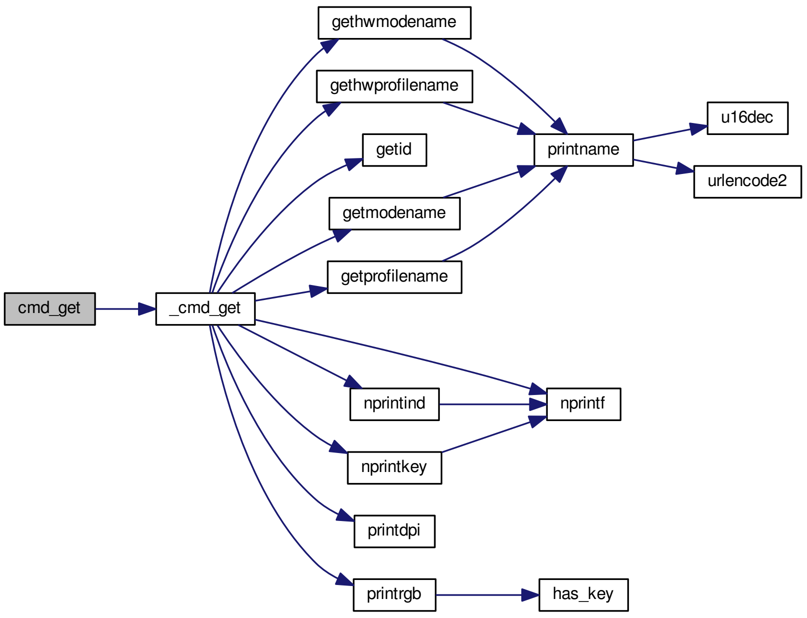
5.30.1.1 void cmd_get (usbdevice ∗ *kb,* usbmode ∗ *mode,* int *nnumber,* int *dummy,* const char ∗ *setting*)

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:



5.30.1.2 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.

```
                                                                                                {
    if(keyindex >= N_KEYS_INPUT)
        return;
    pthread_mutex_lock(imutex(kb));
    if(!strcmp(toggle, "on") || *toggle == 0)
        SET_KEYBIT(mode->notify[nnumber], keyindex);
    else if(!strcmp(toggle, "off"))
        CLEAR_KEYBIT(mode->notify[nnumber], keyindex);
    pthread_mutex_unlock(imutex(kb));
}
```

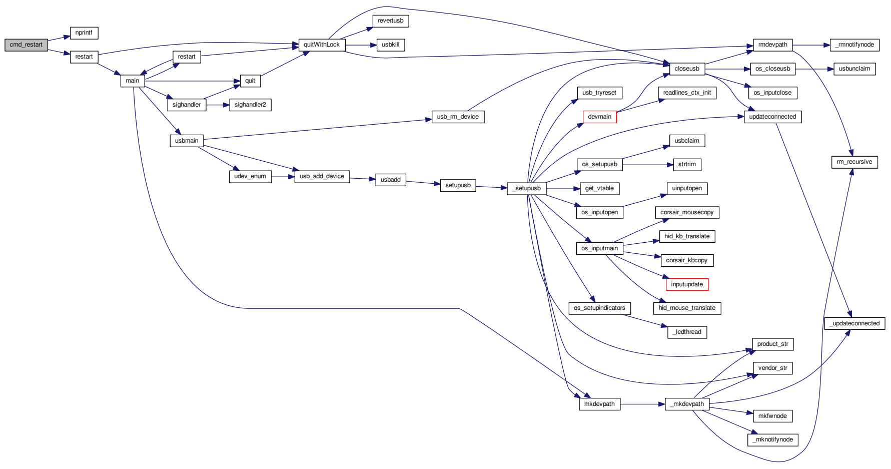
5.30.1.3 void cmd_restart (**usbdevice** ∗ *kb,* **usbmode** ∗ *mode,* int *nnumber,* int *dummy,* const char ∗ *content*)

Definition at line 223 of file notify.c.

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

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

Here is the call graph for this function:



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

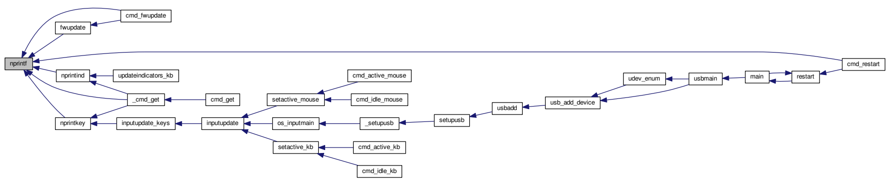
Definition at line 8 of file notify.c.

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

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

```
                                                                                    {
   if(!kb)
        return;
    usbprofile* profile = kb->profile;
    va_list va_args;
    int fifo;
    if(nodenumber >= 0){
        // If node number was given, print to that node (if open)
        if((fifo = kb->outfifo[nodenumber] - 1) != -1){
            va_start(va_args, format);
            if(mode)
                dprintf(fifo, "mode %d ", INDEX_OF(mode, profile->mode) + 1);
            vdprintf(fifo, format, va_args);
        }
        return;
    }
    // Otherwise, print to all nodes
    for(int i = 0; i < OUTFIFO_MAX; i++){
        if((fifo = kb->outfifo[i] - 1) != -1){
            va_start(va_args, format);
            if(mode)
                dprintf(fifo, "mode %d ", INDEX_OF(mode, profile->mode) + 1);
            vdprintf(fifo, format, va_args);
        }
    }
}
```

Here is the caller graph for this function:



5.30.1.5 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().

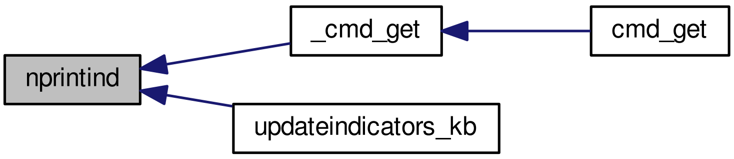
Referenced by _cmd_get(), and updateindicators_kb().

```
                                                                      {
    const char* name = 0;
    switch(led){
    case I_NUM:
        name = "num";
        break;
    case I_CAPS:
        name = "caps";
        break;
    case I_SCROLL:
        name = "scroll";
        break;
    default:
        return;
    }
    nprintf(kb, nnumber, 0, "i %c%s\n", on ? '+' : '-', name);
}
```

Here is the call graph for this function:

nprintind
nprintf

Here is the caller graph for this function:



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

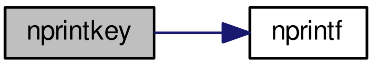
Definition at line 35 of file notify.c.

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

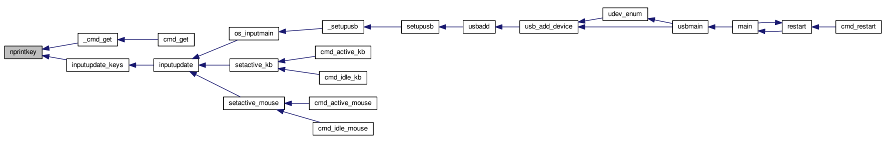
Referenced by _cmd_get(), and inputupdate_keys().

```
                                                                    {
      const key* map = keymap + keyindex;
      if(map->name)
          nprintf(kb, nnumber, 0, "key %c%s\n", down ? '+' : '-', map->name);
      else
          nprintf(kb, nnumber, 0, "key %c#%d\n", down ? '+' : '-', keyindex);
  }
```

Here is the call graph for this function:



Here is the caller graph for this function:

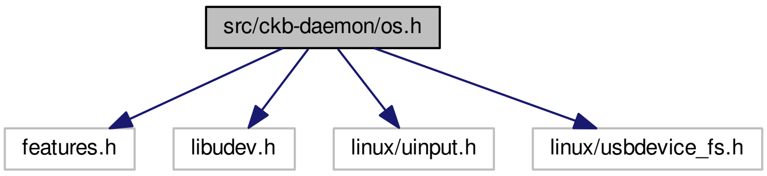


5.31 src/ckb-daemon/os.h File Reference

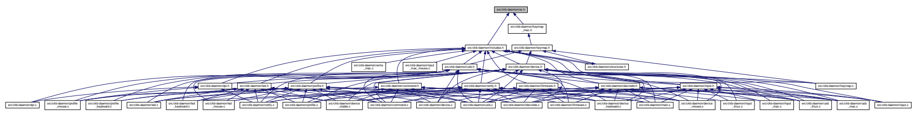
```
#include <features.h>
```

```
#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

5.31.1 Macro Definition Documentation

5.31.1.1 #define _DEFAULT_SOURCE

Definition at line 22 of file os.h.

5.31.1.2 #define _GNU_SOURCE

Definition at line 26 of file os.h.

5.31.1.3 #define euid_guard_start

Definition at line 40 of file os.h.

Referenced by mkdevpath(), mknotifynode(), rmdevpath(), rmnotifynode(), and updateconnected().

5.31.1.4 #define euid_guard_stop

Definition at line 41 of file os.h.

Referenced by mkdevpath(), mknotifynode(), rmdevpath(), rmnotifynode(), and updateconnected().

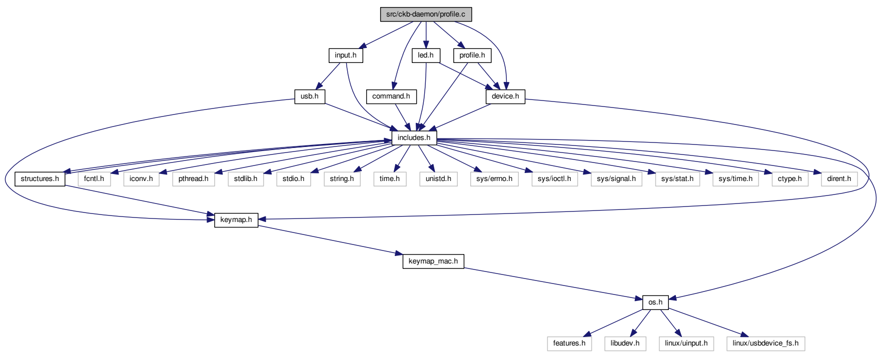
5.31.1.5 #define UINPUT_VERSION 2

Definition at line 35 of file os.h.

5.32 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

5.32.1 Function Documentation

5.32.1.1 static void _freeprofile (usbdevice ∗ *kb*) [static]

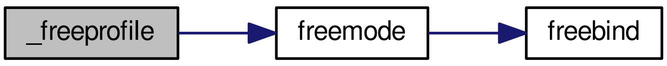
Definition at line 210 of file profile.c.

References freemode(), usbprofile::mode, MODE_COUNT, and usbdevice::profile.

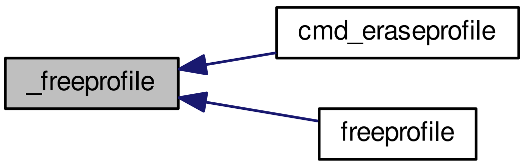
Referenced by cmd_eraseprofile(), and freeprofile().

```
                                          {
    usbprofile* profile = kb->profile;
    if(!profile)
        return;
    // Clear all mode data
    for(int i = 0; i < MODE_COUNT; i++)
        freemode(profile->mode + i);
    free(profile);
    kb->profile = 0;
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.32.1.2 void allocprofile (usbdevice ∗ *kb*)

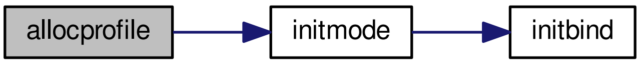
Definition at line 182 of file profile.c.

References usbprofile::currentmode, dpiset::forceupdate, lighting::forceupdate, initmode(), usbprofile::lastdpi, usbprofile::lastlight, usbprofile::mode, MODE_COUNT, and usbdevice::profile.

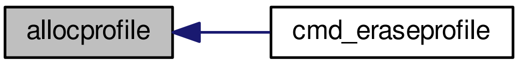
Referenced by cmd_eraseprofile().

```
                                {
    if(kb->profile)
        return;
    usbprofile* profile = kb->profile = calloc(1, sizeof(
  usbprofile));
    for(int i = 0; i < MODE_COUNT; i++)
        initmode(profile->mode + i);
    profile->currentmode = profile->mode;
    profile->lastlight.forceupdate = profile->lastdpi.
  forceupdate = 1;
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



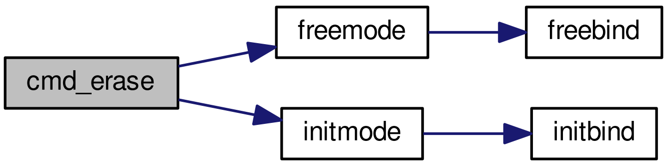
5.32.1.3 void cmd_erase (**usbdevice** ∗ *kb,* **usbmode** ∗ *mode,* int *dummy1,* int *dummy2,* const char ∗ *dummy3*)

Definition at line 203 of file profile.c.

References freemode(), imutex, and initmode().

```
                                                                                  {
    pthread_mutex_lock(imutex(kb));
    freemode(mode);
    initmode(mode);
    pthread_mutex_unlock(imutex(kb));
}
```

Here is the call graph for this function:



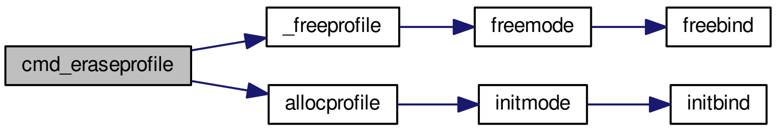
5.32.1.4 void cmd_eraseprofile (usbdevice ∗ *kb,* usbmode ∗ *dummy1,* int *dummy2,* int *dummy3,* const char ∗ *dummy4*)

Definition at line 221 of file profile.c.

References _freeprofile(), allocprofile(), and imutex.

```
                                                                                        {
    pthread_mutex_lock(imutex(kb));
    _freeprofile(kb);
    allocprofile(kb);
    pthread_mutex_unlock(imutex(kb));
}
```

Here is the call graph for this function:



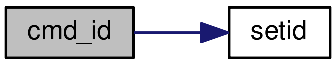
5.32.1.5 void cmd_id (usbdevice ∗ *kb,* usbmode ∗ *mode,* int *dummy1,* int *dummy2,* const char ∗ *id*)

Definition at line 160 of file profile.c.

References usbmode::id, usbid::modified, and setid().

```
                                                                          {
    // ID is either a GUID or an 8-digit hex number
    int newmodified;
    if(!setid(&mode->id, id) && sscanf(id, "%08x", &newmodified) == 1)
        memcpy(mode->id.modified, &newmodified, sizeof(newmodified));
}
```

Here is the call graph for this function:



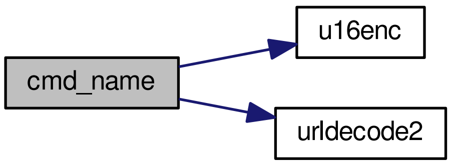
5.32.1.6 void cmd_name (usbdevice ∗ *kb*, usbmode ∗ *mode*, int *dummy1*, int *dummy2*, const char ∗ *name*)

Definition at line 117 of file profile.c.

References MD_NAME_LEN, usbmode::name, u16enc(), and urldecode2().

```
                                                                                     {
    char decoded[strlen(name) + 1];
    urldecode2(decoded, name);
    size_t srclen = strlen(decoded), dstlen = MD_NAME_LEN;
    u16enc(decoded, mode->name, &srclen, &dstlen);
}
```

Here is the call graph for this function:



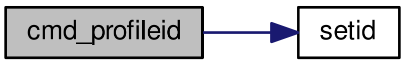
5.32.1.7 void cmd_profileid (usbdevice ∗ *kb*, usbmode ∗ *mode*, int *dummy1*, int *dummy2*, const char ∗ *id*)

Definition at line 167 of file profile.c.

References usbprofile::id, usbid::modified, usbdevice::profile, and setid().

```
                                                                                        {
    usbprofile* profile = kb->profile;
    int newmodified;
    if(!setid(&profile->id, id) && sscanf(id, "%08x", &newmodified) == 1)
        memcpy(profile->id.modified, &newmodified, sizeof(newmodified));

}
```

Here is the call graph for this function:



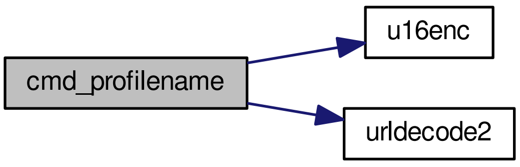
5.32.1.8 void cmd_profilename (**usbdevice** ∗ *kb,* **usbmode** ∗ *dummy1,* int *dummy2,* int *dummy3,* const char ∗ *name*)

Definition at line 124 of file profile.c.

References usbprofile::name, PR_NAME_LEN, usbdevice::profile, u16enc(), and urldecode2().

```
                                                                                     {
    usbprofile* profile = kb->profile;
    char decoded[strlen(name) + 1];
    urldecode2(decoded, name);
    size_t srclen = strlen(decoded), dstlen = PR_NAME_LEN;
    u16enc(decoded, profile->name, &srclen, &dstlen);
}
```

Here is the call graph for this function:



5.32.1.9 static void freemode (**usbmode** ∗ *mode*) [static]

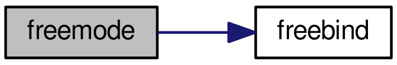
Definition at line 198 of file profile.c.

References usbmode::bind, and freebind().

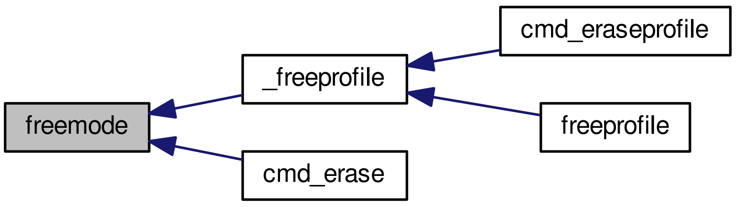
Referenced by _freeprofile(), and cmd_erase().

```
                              {
    freebind(&mode->bind);
    memset(mode, 0, sizeof(*mode));
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



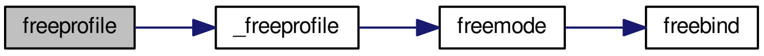
5.32.1.10 void freeprofile (**usbdevice** ∗ *kb*)

Definition at line 228 of file profile.c.

References _freeprofile(), and usbdevice::hw.

```
                             {
    _freeprofile(kb);
    // Also free HW profile
    free(kb->hw);
    kb->hw = 0;
}
```

Here is the call graph for this function:



5.32.1.11 char∗ gethwmodename (**hwprofile** ∗ *profile,* int *index*)

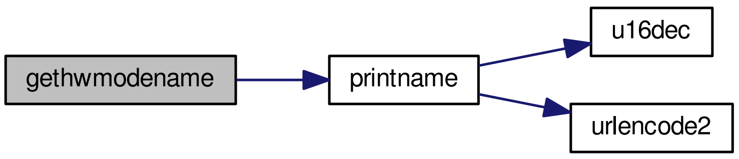
Definition at line 152 of file profile.c.

References MD_NAME_LEN, hwprofile::name, and printname().

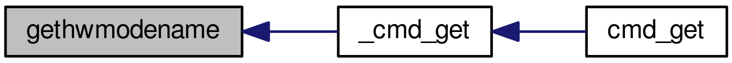
Referenced by _cmd_get().

```
                                        {
    return printname(profile->name[index + 1], MD_NAME_LEN);
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.32.1.12 char∗ gethwprofilename (hwprofile ∗ *profile*)

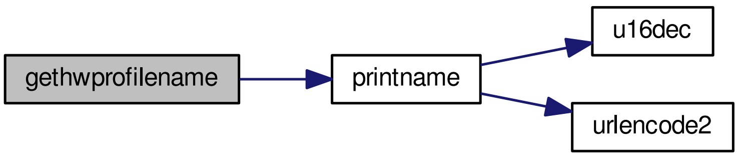
Definition at line 156 of file profile.c.

References MD_NAME_LEN, hwprofile::name, and printname().

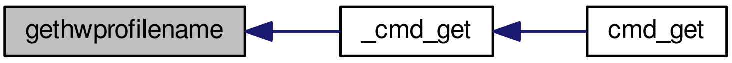
Referenced by _cmd_get().

```
                                  {
    return printname(profile->name[0], MD_NAME_LEN);
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.32.1.13 char∗ getid (usbid ∗ *id*)

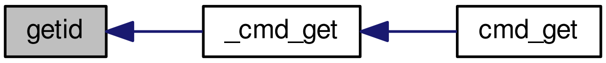
Definition at line 79 of file profile.c.

References usbid::guid.

Referenced by _cmd_get().

```
                       {
    int32_t data1;
    int16_t data2, data3, data4a;
    char data4b[6];
    memcpy(&data1, id->guid + 0x0, 4);
    memcpy(&data2, id->guid + 0x4, 2);
    memcpy(&data3, id->guid + 0x6, 2);
    memcpy(&data4a, id->guid + 0x8, 2);
    memcpy(data4b, id->guid + 0xA, 6);
    char* guid = malloc(39);
    snprintf(guid, 39, "{%08X-%04hX-%04hX-%04hX-%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX}",
             data1, data2, data3, data4a, data4b[0], data4b[1], data4b[2], data4b[3], data4b[4], data4b[5])
   ;
    return guid;
}
```

Here is the caller graph for this function:



5.32.1.14 char∗ getmodename (usbmode ∗ *mode*)

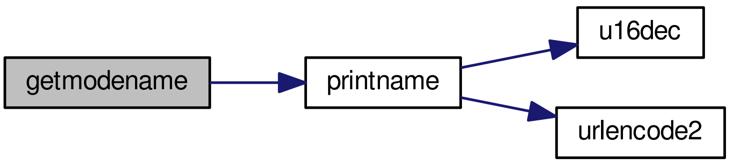
Definition at line 144 of file profile.c.

References MD_NAME_LEN, usbmode::name, and printname().

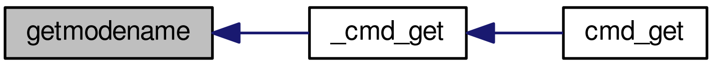
Referenced by _cmd_get().

```
                                {
    return printname(mode->name, MD_NAME_LEN);
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.32.1.15 char∗ getprofilename (usbprofile ∗ *profile*)

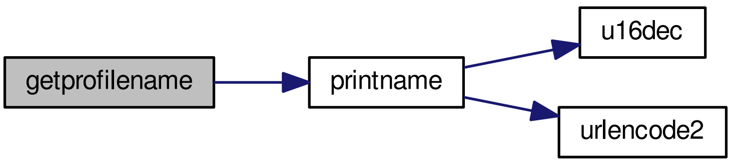
Definition at line 148 of file profile.c.

References usbprofile::name, PR_NAME_LEN, and printname().

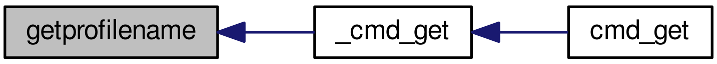
Referenced by _cmd_get().

```
                                           {
    return printname(profile->name, PR_NAME_LEN);
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.32.1.16 void hwtonative (**usbprofile** ∗ *profile,* **hwprofile** ∗ *hw,* int *modecount*)

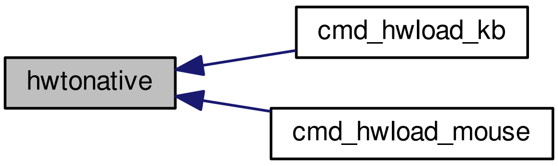
Definition at line 235 of file profile.c.

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().

```
                                                                  {
    // Copy the profile name and ID
    memcpy(profile->name, hw->name[0], PR_NAME_LEN * 2);
    memcpy(&profile->id, hw->id, sizeof(usbid));
    // Copy the mode settings
    for(int i = 0; i < modecount; i++){
        usbmode* mode = profile->mode + i;
        memcpy(mode->name, hw->name[i + 1], MD_NAME_LEN * 2);
        memcpy(&mode->id, hw->id + i + 1, sizeof(usbid));
        memcpy(&mode->light, hw->light + i, sizeof(lighting));
        memcpy(&mode->dpi, hw->dpi + i, sizeof(dpiset));
        // Set a force update on the light/DPI since they've been overwritten
        mode->light.forceupdate = mode->dpi.forceupdate = 1;
    }
    profile->lastlight.forceupdate = profile->lastdpi.
  forceupdate = 1;
}
```

Here is the caller graph for this function:



5.32.1.17 static void initmode (**usbmode** ∗ *mode*) [static]

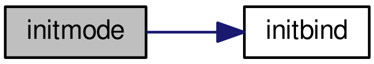
Definition at line 175 of file profile.c.

References usbmode::bind, usbmode::dpi, dpiset::forceupdate, lighting::forceupdate, initbind(), and usbmode::light.

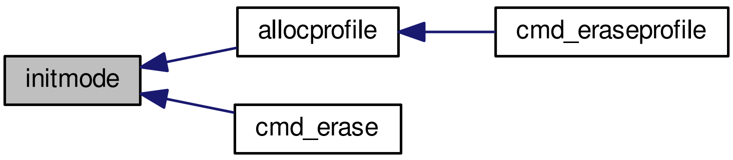
Referenced by allocprofile(), and cmd_erase().

```
                                  {
    memset(mode, 0, sizeof(*mode));
    mode->light.forceupdate = 1;
    mode->dpi.forceupdate = 1;
    initbind(&mode->bind);
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.32.1.18 int loadprofile (**usbdevice** ∗ *kb*)

Definition at line 192 of file profile.c.

References hwloadprofile.

```
                             {
    if(hwloadprofile(kb, 1))
        return -1;
    return 0;
}
```

5.32.1.19 void nativetohw (**usbprofile** ∗ *profile,* **hwprofile** ∗ *hw,* int *modecount*)

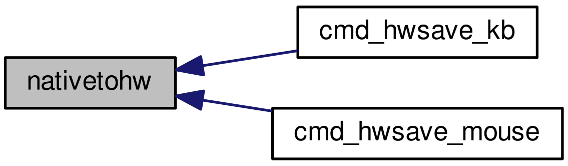
Definition at line 252 of file profile.c.

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().

```
                                                                   {
    // Copy name and ID
    memcpy(hw->name[0], profile->name, PR_NAME_LEN * 2);
    memcpy(hw->id, &profile->id, sizeof(usbid));
    // Copy the mode settings
    for(int i = 0; i < modecount; i++){
        usbmode* mode = profile->mode + i;
        memcpy(hw->name[i + 1], mode->name, MD_NAME_LEN * 2);
        memcpy(hw->id + i + 1, &mode->id, sizeof(usbid));
        memcpy(hw->light + i, &mode->light, sizeof(lighting));
        memcpy(hw->dpi + i, &mode->dpi, sizeof(dpiset));
    }
}
```

Here is the caller graph for this function:



5.32.1.20 char* printname (ushort * *name,* int *length*)

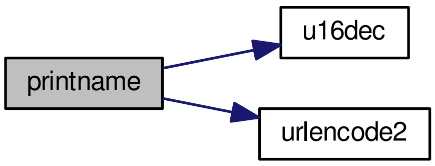
Definition at line 132 of file profile.c.

References u16dec(), and urlencode2().

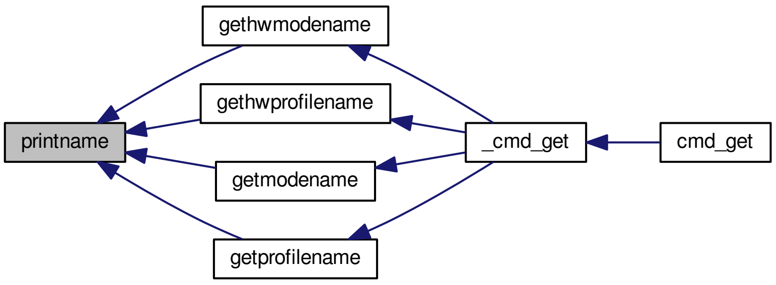
Referenced by gethwmodename(), gethwprofilename(), getmodename(), and getprofilename().

```
                                    {
    // Convert the name to UTF-8
    char* buffer = calloc(1, length * 4 - 3);
    size_t srclen = length, dstlen = length * 4 - 4;
    u16dec(name, buffer, &srclen, &dstlen);
    // URL-encode it
    char* buffer2 = malloc(strlen(buffer) * 3 + 1);
    urlencode2(buffer2, buffer);
    free(buffer);
    return buffer2;
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.32.1.21 int setid (usbid ∗ *id,* const char ∗ *guid*)

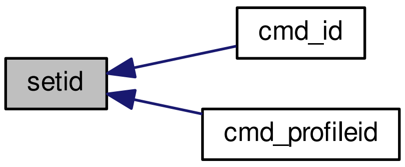
Definition at line 64 of file profile.c.

References usbid::guid.

Referenced by cmd_id(), and cmd_profileid().

```
                                       {
    int32_t data1;
    int16_t data2, data3, data4a;
    char data4b[6];
    if(sscanf(guid, "{%08X-%04hX-%04hX-%04hX-%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX}",
              &data1, &data2, &data3, &data4a, data4b, data4b + 1, data4b + 2, data4b + 3, data4b + 4,
   data4b + 5) != 10)
        return 0;
    memcpy(id->guid + 0x0, &data1, 4);
    memcpy(id->guid + 0x4, &data2, 2);
    memcpy(id->guid + 0x6, &data3, 2);
    memcpy(id->guid + 0x8, &data4a, 2);
    memcpy(id->guid + 0xA, data4b, 6);
    return 1;
}
```

Here is the caller graph for this function:



5.32.1.22 void u16dec (ushort ∗ *in,* char ∗ *out,* size_t ∗ *srclen,* size_t ∗ *dstlen*)

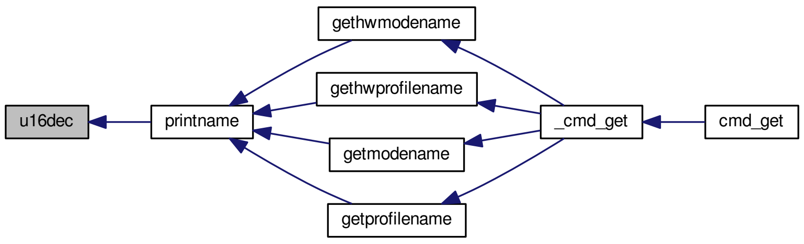
Definition at line 105 of file profile.c.

References utf16to8.

Referenced by printname().

```
                                                                  {
    if(!utf16to8)
        utf16to8 = iconv_open("UTF-8", "UTF-16LE");
    size_t srclen2 = 0, srclenmax = *srclen;
    for(; srclen2 < srclenmax; srclen2++){
        if(!in[srclen2])
            break;
    }
    *srclen = srclen2 * 2;
    iconv(utf16to8, (char**)&in, srclen, &out, dstlen);
}
```

Here is the caller graph for this function:



5.32.1.23 void u16enc (char * *in,* ushort * *out,* size_t * *srclen,* size_t * *dstlen*)

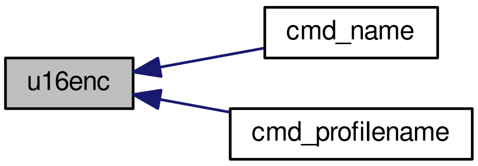
Definition at line 97 of file profile.c.

References utf8to16.

Referenced by cmd_name(), and cmd_profilename().

```
                                                                  {
    if(!utf8to16)
        utf8to16 = iconv_open("UTF-16LE", "UTF-8");
    memset(out, 0, *dstlen * 2);
    *dstlen = *dstlen * 2 - 2;
    iconv(utf8to16, &in, srclen, (char**)&out, dstlen);
}
```

Here is the caller graph for this function:



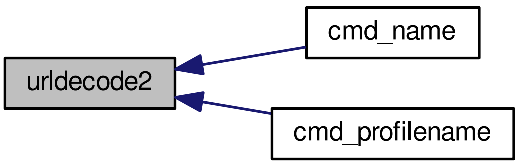
5.32.1.24 void urldecode2 (char ∗ *dst,* const char ∗ *src*)

Definition at line 8 of file profile.c.

Referenced by cmd_name(), and cmd_profilename().

```
                                              {
    char a, b;
     char s;
     while((s = *src)){
         if((s == '%') &&
                 ((a = src[1]) && (b = src[2])) &&
                 (isxdigit(a) && isxdigit(b))){
             if(a >= 'a')
                 a -= 'a'-'A';
             if(a >= 'A')
                 a -= 'A' - 10;
             else
                 a -= '0';
             if(b >= 'a')
                 b -= 'a'-'A';
             if(b >= 'A')
                 b -= 'A' - 10;
             else
                 b -= '0';
             *dst++ = 16 * a + b;
             src += 3;
         } else {
             *dst++ = s;
             src++;
         }
     }
     *dst = '\0';
 }
```

Here is the caller graph for this function:



5.32.1.25 void urlencode2 (char ∗ *dst,* const char ∗ *src*)

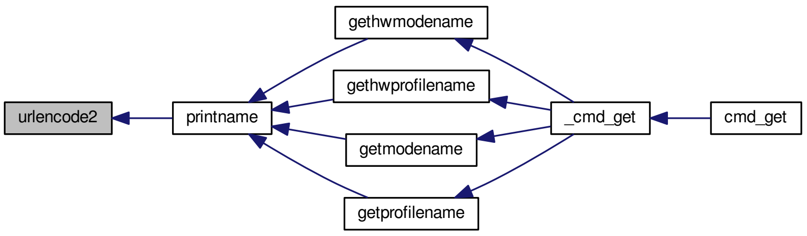
Definition at line 37 of file profile.c.

Referenced by printname().

```
                                               {
     char s;
     while((s = *src++)){
         if(s <= ',' || s == '/' ||
                 (s >= ':' && s <= '@') ||
                 s == '[' || s == ']' ||
                 s >= 0x7F){
             char a = s >> 4, b = s & 0xF;
             if(a >= 10)
                 a += 'A' - 10;
             else
                 a += '0';
             if(b >= 10)
                 b += 'A' - 10;
             else
```

```
                b += '0';
            dst[0] = '%';
            dst[1] = a;
            dst[2] = b;
            dst += 3;
        } else
            *dst++ = s;
    }
    *dst = '\0';
}
```

Here is the caller graph for this function:



5.32.2 Variable Documentation

5.32.2.1 iconv_t utf16to8 = 0 `[static]`

Definition at line 95 of file profile.c.

Referenced by u16dec().

5.32.2.2 iconv_t utf8to16 = 0 `[static]`

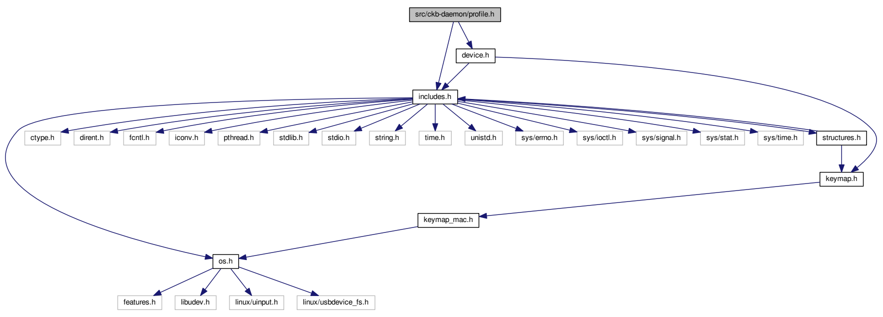
Definition at line 95 of file profile.c.

Referenced by u16enc().

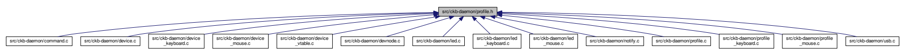
5.33 src/ckb-daemon/profile.h File Reference

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

Include dependency graph for profile.h:



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


src/ckb-daemon/profile.h
src/ckb-daemon/command.c
src/ckb-daemon/device.c
src/ckb-daemon/device_keyboard.c
src/ckb-daemon/device_mouse.c
src/ckb-daemon/device_vtable.c
src/ckb-daemon/devnode.c
src/ckb-daemon/led.c
src/ckb-daemon/led_keyboard.c
src/ckb-daemon/led_mouse.c
src/ckb-daemon/notify.c
src/ckb-daemon/profile.c
src/ckb-daemon/profile_keyboard.c
src/ckb-daemon/profile_mouse.c
src/ckb-daemon/usb.c

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)

5.33.1 Macro Definition Documentation

5.33.1.1 #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().

5.33.2 Function Documentation

5.33.2.1 void allocprofile (usbdevice ∗ *kb*)

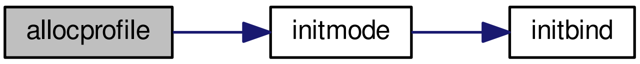
Definition at line 182 of file profile.c.

References usbprofile::currentmode, dpiset::forceupdate, lighting::forceupdate, initmode(), usbprofile::lastdpi, usbprofile::lastlight, usbprofile::mode, MODE_COUNT, and usbdevice::profile.

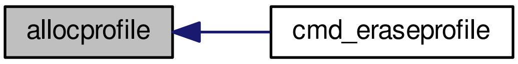
Referenced by cmd_eraseprofile().

```
                                {
    if(kb->profile)
        return;
    usbprofile* profile = kb->profile = calloc(1, sizeof(
  usbprofile));
    for(int i = 0; i < MODE_COUNT; i++)
        initmode(profile->mode + i);
    profile->currentmode = profile->mode;
    profile->lastlight.forceupdate = profile->lastdpi.
  forceupdate = 1;
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



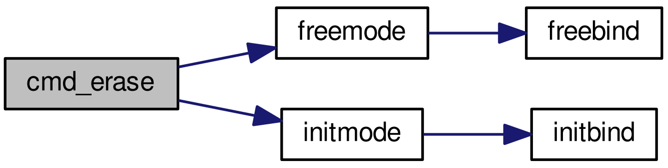
5.33.2.2 void cmd_erase (**usbdevice** ∗ *kb,* **usbmode** ∗ *mode,* int *dummy1,* int *dummy2,* const char ∗ *dummy3*)

Definition at line 203 of file profile.c.

References freemode(), imutex, and initmode().

```
                                                                                  {
    pthread_mutex_lock(imutex(kb));
    freemode(mode);
    initmode(mode);
    pthread_mutex_unlock(imutex(kb));
}
```

Here is the call graph for this function:



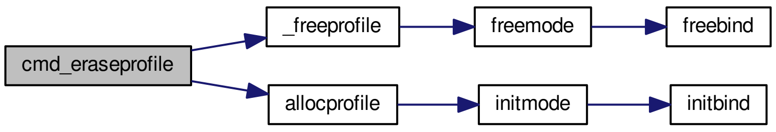
5.33.2.3 void cmd_eraseprofile (usbdevice ∗ *kb*, usbmode ∗ *dummy1*, int *dummy2*, int *dummy3*, const char ∗ *dummy4*)

Definition at line 221 of file profile.c.

References _freeprofile(), allocprofile(), and imutex.

```
                                                                                                                   {
    pthread_mutex_lock(imutex(kb));
    _freeprofile(kb);
    allocprofile(kb);
    pthread_mutex_unlock(imutex(kb));
}
```

Here is the call graph for this function:



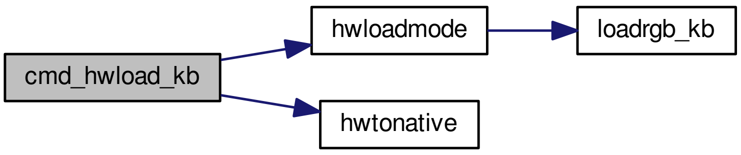
5.33.2.4 int cmd_hwload_kb (usbdevice ∗ *kb*, usbmode ∗ *dummy1*, int *dummy2*, int *apply*, const char ∗ *dummy3*)

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.

```
                                                                                                           {
    DELAY_LONG(kb);
    hwprofile* hw = calloc(1, sizeof(hwprofile));
    // Ask for profile and mode IDs
    uchar data_pkt[2][MSG_SIZE] = {
        { 0x0e, 0x15, 0x01, 0 },
        { 0x0e, 0x16, 0x01, 0 }
    };
    uchar in_pkt[MSG_SIZE];
    int modes = (IS_K95(kb) ? HWMODE_K95 : HWMODE_K70);
    for(int i = 0; i <= modes; i++){
        data_pkt[0][3] = i;
        if(!usbrecv(kb, data_pkt[0], in_pkt)){
            free(hw);
            return -1;
        }
        memcpy(hw->id + i, in_pkt + 4, sizeof(usbid));
    }
    // Ask for profile name
    if(!usbrecv(kb, data_pkt[1], in_pkt)){
        free(hw);
        return -1;
    }
    memcpy(hw->name[0], in_pkt + 4, PR_NAME_LEN * 2);
    // Load modes
    for(int i = 0; i < modes; i++){
        if(hwloadmode(kb, hw, i)){
            free(hw);
            return -1;
        }
    }
    // Make the profile active (if requested)
    if(apply)
        hwtonative(kb->profile, hw, modes);
    // Free the existing profile (if any)
    free(kb->hw);
    kb->hw = hw;
    DELAY_LONG(kb);
    return 0;
}
```

Here is the call graph for this function:



5.33.2.5 int cmd_hwload_mouse (usbdevice ∗ *kb,* usbmode ∗ *dummy1,* int *dummy2,* int *apply,* const char ∗ *dummy3*)

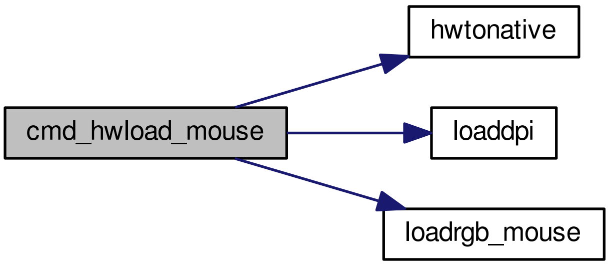
Definition at line 6 of file profile_mouse.c.

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.

```
                                                                                           {
     DELAY_LONG(kb);
     hwprofile* hw = calloc(1, sizeof(hwprofile));
     // Ask for profile and mode IDs
      uchar data_pkt[2][MSG_SIZE] = {
          { 0x0e, 0x15, 0x01, 0 },
          { 0x0e, 0x16, 0x01, 0 }
      };
      uchar in_pkt[MSG_SIZE];
      for(int i = 0; i <= 1; i++){
          data_pkt[0][3] = i;
          if(!usbrecv(kb, data_pkt[0], in_pkt)){
              free(hw);
              return -1;
          }
          memcpy(hw->id + i, in_pkt + 4, sizeof(usbid));
      }
      // Ask for profile and mode names
      for(int i = 0; i <= 1; i++){
          data_pkt[1][3] = i;
          if(!usbrecv(kb, data_pkt[1],in_pkt)){
              free(hw);
              return -1;
          }
```

```
        memcpy(hw->name[i], in_pkt + 4, PR_NAME_LEN * 2);
    }

    // Load the RGB and DPI settings
    if(loadrgb_mouse(kb, hw->light, 0)
            || loaddpi(kb, hw->dpi, hw->light)){
        free(hw);
        return -1;
    }

    // Make the profile active (if requested)
    if(apply)
        hwtonative(kb->profile, hw, 1);
    // Free the existing profile (if any)
    free(kb->hw);
    kb->hw = hw;
    DELAY_LONG(kb);
    return 0;
}
```

Here is the call graph for this function:



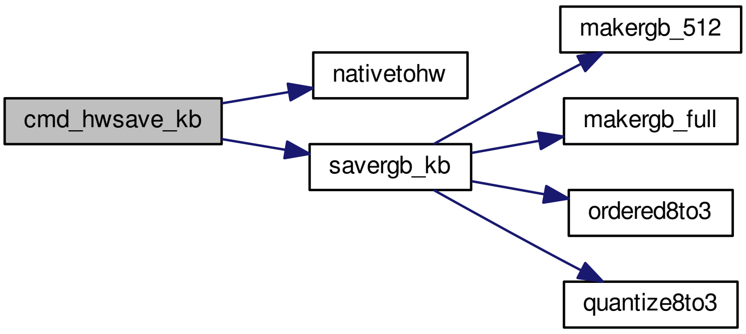
5.33.2.6 int cmd_hwsave_kb (**usbdevice** ∗ *kb,* **usbmode** ∗ *dummy1,* int *dummy2,* int *dummy3,* const char ∗ *dummy4*)

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.

```
                                                                                         {
    DELAY_LONG(kb);
    hwprofile* hw = kb->hw;
    if(!hw)
        hw = kb->hw = calloc(1, sizeof(hwprofile));
    int modes = (IS_K95(kb) ? HWMODE_K95 : HWMODE_K70);
    nativetohw(kb->profile, hw, modes);
    // Save the profile and mode names
    uchar data_pkt[2][MSG_SIZE] = {
        { 0x07, 0x16, 0x01, 0 },
        { 0x07, 0x15, 0x01, 0 },
    };
    // Save the mode names
    for(int i = 0; i <= modes; i++){
        data_pkt[0][3] = i;
        memcpy(data_pkt[0] + 4, hw->name[i], MD_NAME_LEN * 2);
        if(!usbsend(kb, data_pkt[0], 1))
            return -1;
    }
    // Save the IDs
    for(int i = 0; i <= modes; i++){
        data_pkt[1][3] = i;
        memcpy(data_pkt[1] + 4, hw->id + i, sizeof(usbid));
        if(!usbsend(kb, data_pkt[1], 1))
            return -1;
    }
    // Save the RGB data
    for(int i = 0; i < modes; i++){
        if(savergb_kb(kb, hw->light + i, i))
            return -1;
    }
    DELAY_LONG(kb);
    return 0;
}
```

Here is the call graph for this function:



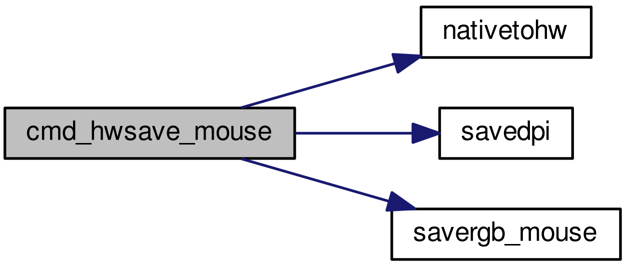
5.33.2.7 int cmd_hwsave_mouse (usbdevice ∗ *kb,* usbmode ∗ *dummy1,* int *dummy2,* int *dummy3,* const char ∗ *dummy4*)

Definition at line 50 of file profile_mouse.c.

References DELAY_LONG, hwprofile::dpi, usbdevice::hw, hwprofile::id, hwprofile::light, MD_NAME_LEN, MSG_S-IZE, hwprofile::name, nativetohw(), usbdevice::profile, savedpi(), savergb_mouse(), and usbsend.

```
                                                                                               {
    DELAY_LONG(kb);
    hwprofile* hw = kb->hw;
    if(!hw)
        hw = kb->hw = calloc(1, sizeof(hwprofile));
    nativetohw(kb->profile, hw, 1);
    // Save the profile and mode names
    uchar data_pkt[2][MSG_SIZE] = {
        { 0x07, 0x16, 0x01, 0 },
        { 0x07, 0x15, 0x01, 0 },
    };
    for(int i = 0; i <= 1; i++){
        data_pkt[0][3] = i;
        memcpy(data_pkt[0] + 4, hw->name[i], MD_NAME_LEN * 2);
        if(!usbsend(kb, data_pkt[0], 1))
            return -1;
    }
    // Save the IDs
    for(int i = 0; i <= 1; i++){
        data_pkt[1][3] = i;
        memcpy(data_pkt[1] + 4, hw->id + i, sizeof(usbid));
        if(!usbsend(kb, data_pkt[1], 1))
            return -1;
    }
    // Save the RGB data for the non-DPI zones
    if(savergb_mouse(kb, hw->light, 0))
        return -1;
    // Save the DPI data (also saves RGB for those states)
    if(savedpi(kb, hw->dpi, hw->light))
        return -1;
    DELAY_LONG(kb);
    return 0;
}
```

Here is the call graph for this function:



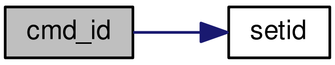
5.33.2.8 void cmd_id (usbdevice ∗ *kb,* usbmode ∗ *mode,* int *dummy1,* int *dummy2,* const char ∗ *id*)

Definition at line 160 of file profile.c.

References usbmode::id, usbid::modified, and setid().

```
                                                                                          {
    // ID is either a GUID or an 8-digit hex number
    int newmodified;
    if(!setid(&mode->id, id) && sscanf(id, "%08x", &newmodified) == 1)
        memcpy(mode->id.modified, &newmodified, sizeof(newmodified));
}
```

Here is the call graph for this function:



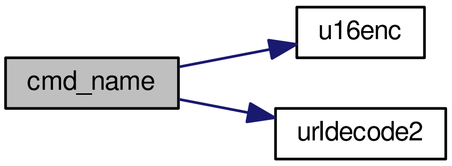
5.33.2.9 void cmd_name (usbdevice ∗ *kb,* usbmode ∗ *mode,* int *dummy1,* int *dummy2,* const char ∗ *name*)

Definition at line 117 of file profile.c.

References MD_NAME_LEN, usbmode::name, u16enc(), and urldecode2().

```
                                                                                              {
    char decoded[strlen(name) + 1];
    urldecode2(decoded, name);
    size_t srclen = strlen(decoded), dstlen = MD_NAME_LEN;
    u16enc(decoded, mode->name, &srclen, &dstlen);
}
```

Here is the call graph for this function:



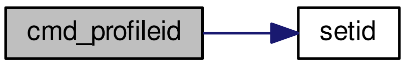
5.33.2.10 void cmd_profileid (usbdevice ∗ *kb,* usbmode ∗ *mode,* int *dummy1,* int *dummy2,* const char ∗ *id*)

Definition at line 167 of file profile.c.

References usbprofile::id, usbid::modified, usbdevice::profile, and setid().

```
                                                                                    {
    usbprofile* profile = kb->profile;
    int newmodified;
    if(!setid(&profile->id, id) && sscanf(id, "%08x", &newmodified) == 1)
        memcpy(profile->id.modified, &newmodified, sizeof(newmodified));

}
```

Here is the call graph for this function:



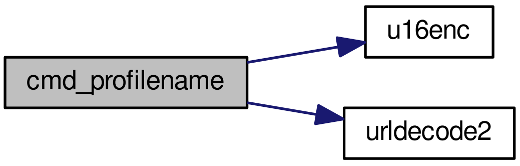
5.33.2.11 void cmd_profilename (usbdevice ∗ *kb,* usbmode ∗ *dummy1,* int *dummy2,* int *dummy3,* const char ∗ *name*)

Definition at line 124 of file profile.c.

References usbprofile::name, PR_NAME_LEN, usbdevice::profile, u16enc(), and urldecode2().

```
                                                                                         {
    usbprofile* profile = kb->profile;
    char decoded[strlen(name) + 1];
    urldecode2(decoded, name);
    size_t srclen = strlen(decoded), dstlen = PR_NAME_LEN;
    u16enc(decoded, profile->name, &srclen, &dstlen);
}
```

Here is the call graph for this function:



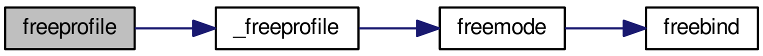
5.33.2.12 void freeprofile (usbdevice ∗ *kb*)

Definition at line 228 of file profile.c.

References _freeprofile(), and usbdevice::hw.

```
                                      {
    _freeprofile(kb);
    // Also free HW profile
    free(kb->hw);
    kb->hw = 0;
}
```

Here is the call graph for this function:



5.33.2.13 char∗ gethwmodename (hwprofile ∗ *profile,* int *index*)

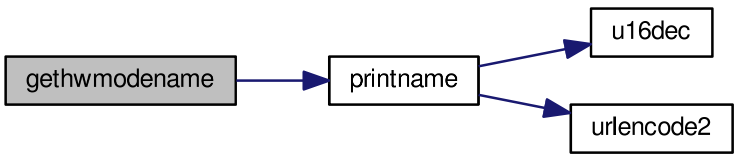
Definition at line 152 of file profile.c.

References MD_NAME_LEN, hwprofile::name, and printname().

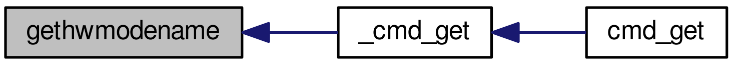
Referenced by _cmd_get().

```
                                                  {
    return printname(profile->name[index + 1], MD_NAME_LEN);
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.33.2.14 char∗ gethwprofilename (**hwprofile** ∗ *profile*)

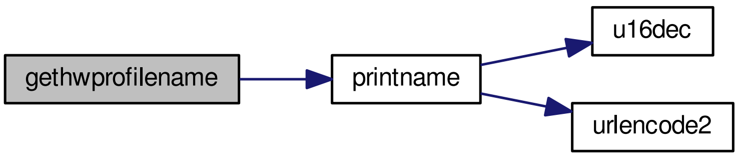
Definition at line 156 of file profile.c.

References MD_NAME_LEN, hwprofile::name, and printname().

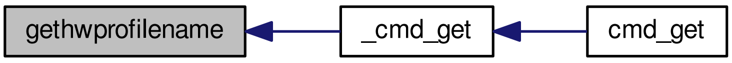
Referenced by _cmd_get().

```
                                                {
    return printname(profile->name[0], MD_NAME_LEN);
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.33.2.15 char∗ getid (usbid ∗ *id*)

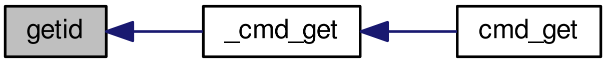
Definition at line 79 of file profile.c.

References usbid::guid.

Referenced by _cmd_get().

```
                          {
    int32_t data1;
    int16_t data2, data3, data4a;
    char data4b[6];
    memcpy(&data1, id->guid + 0x0, 4);
    memcpy(&data2, id->guid + 0x4, 2);
    memcpy(&data3, id->guid + 0x6, 2);
    memcpy(&data4a, id->guid + 0x8, 2);
    memcpy(data4b, id->guid + 0xA, 6);
    char* guid = malloc(39);
    snprintf(guid, 39, "{%08X-%04hX-%04hX-%04hX-%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX}",
             data1, data2, data3, data4a, data4b[0], data4b[1], data4b[2], data4b[3], data4b[4], data4b[5])
   ;
    return guid;
}
```

Here is the caller graph for this function:



5.33.2.16 char∗ getmodename (usbmode ∗ *mode*)

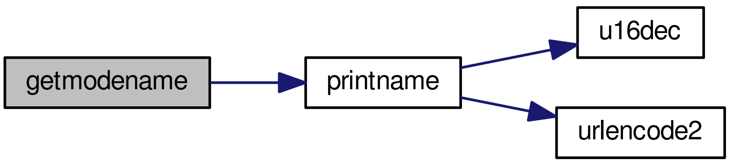
Definition at line 144 of file profile.c.

References MD_NAME_LEN, usbmode::name, and printname().

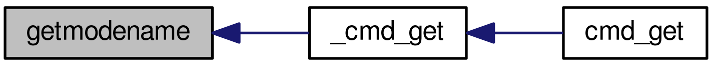
Referenced by _cmd_get().

```
                                  {
    return printname(mode->name, MD_NAME_LEN);
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.33.2.17 char∗ getprofilename (usbprofile ∗ *profile*)

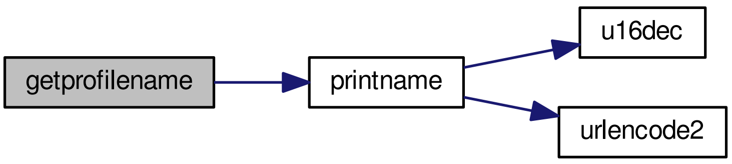
Definition at line 148 of file profile.c.

References usbprofile::name, PR_NAME_LEN, and printname().

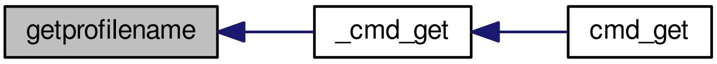
Referenced by _cmd_get().

```
                                        {
    return printname(profile->name, PR_NAME_LEN);
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.33.2.18 void hwtonative (**usbprofile** ∗ *profile,* **hwprofile** ∗ *hw,* int *modecount*)

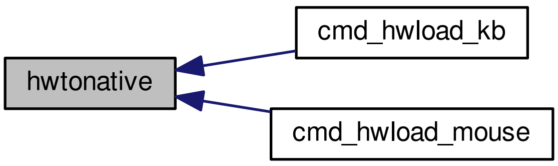
Definition at line 235 of file profile.c.

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().

```
                                                                  {
    // Copy the profile name and ID
    memcpy(profile->name, hw->name[0], PR_NAME_LEN * 2);
    memcpy(&profile->id, hw->id, sizeof(usbid));
    // Copy the mode settings
    for(int i = 0; i < modecount; i++){
        usbmode* mode = profile->mode + i;
        memcpy(mode->name, hw->name[i + 1], MD_NAME_LEN * 2);
        memcpy(&mode->id, hw->id + i + 1, sizeof(usbid));
        memcpy(&mode->light, hw->light + i, sizeof(lighting));
        memcpy(&mode->dpi, hw->dpi + i, sizeof(dpiset));
        // Set a force update on the light/DPI since they've been overwritten
        mode->light.forceupdate = mode->dpi.forceupdate = 1;
    }
    profile->lastlight.forceupdate = profile->lastdpi.
  forceupdate = 1;
}
```

Here is the caller graph for this function:



5.33.2.19 int loadprofile (**usbdevice** ∗ *kb*)

Definition at line 192 of file profile.c.

References hwloadprofile.

```
                            {
    if(hwloadprofile(kb, 1))
        return -1;
    return 0;
}
```

5.33.2.20 void nativetohw (usbprofile ∗ *profile,* hwprofile ∗ *hw,* int *modecount*)

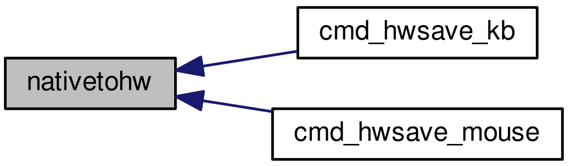
Definition at line 252 of file profile.c.

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().

```
                                                                  {
    // Copy name and ID
    memcpy(hw->name[0], profile->name, PR_NAME_LEN * 2);
    memcpy(hw->id, &profile->id, sizeof(usbid));
    // Copy the mode settings
    for(int i = 0; i < modecount; i++){
        usbmode* mode = profile->mode + i;
        memcpy(hw->name[i + 1], mode->name, MD_NAME_LEN * 2);
        memcpy(hw->id + i + 1, &mode->id, sizeof(usbid));
        memcpy(hw->light + i, &mode->light, sizeof(lighting));
        memcpy(hw->dpi + i, &mode->dpi, sizeof(dpiset));
    }
}
```

Here is the caller graph for this function:



5.33.2.21 int setid (usbid ∗ *id,* const char ∗ *guid*)

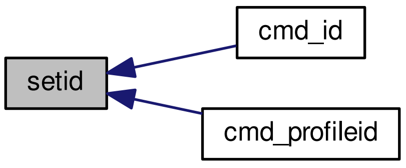
Definition at line 64 of file profile.c.

References usbid::guid.

Referenced by cmd_id(), and cmd_profileid().

```
                                       {
    int32_t data1;
    int16_t data2, data3, data4a;
    char data4b[6];
    if(sscanf(guid, "{%08X-%04hX-%04hX-%04hX-%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX}",
              &data1, &data2, &data3, &data4a, data4b, data4b + 1, data4b + 2, data4b + 3, data4b + 4,
      data4b + 5) != 10)
        return 0;
    memcpy(id->guid + 0x0, &data1, 4);
    memcpy(id->guid + 0x4, &data2, 2);
    memcpy(id->guid + 0x6, &data3, 2);
    memcpy(id->guid + 0x8, &data4a, 2);
    memcpy(id->guid + 0xA, data4b, 6);
    return 1;
}
```

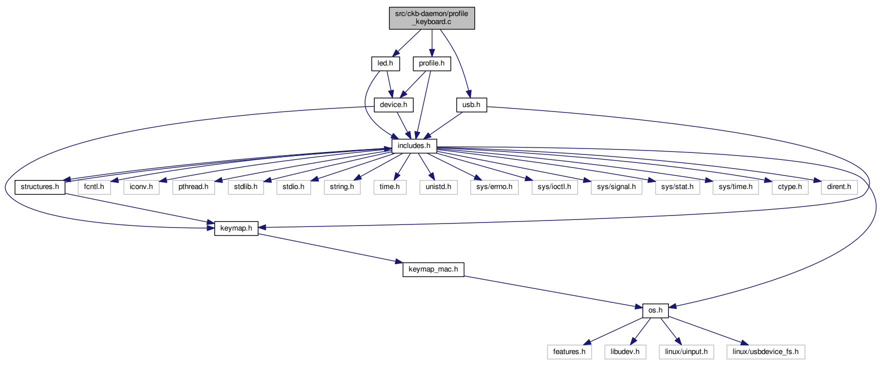
Here is the caller graph for this function:



5.34 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)

5.34.1 Function Documentation

5.34.1.1 int cmd_hwload_kb (usbdevice ∗ *kb,* usbmode ∗ *dummy1,* int *dummy2,* int *apply,* const char ∗ *dummy3*)

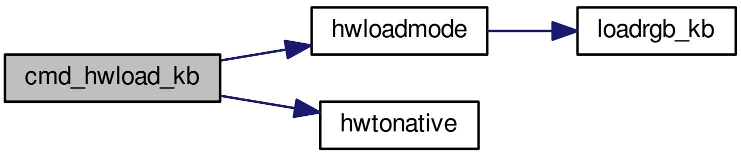
Definition at line 16 of file profile_keyboard.c.

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
```

```
    uchar data_pkt[2][MSG_SIZE] = {
        { 0x0e, 0x15, 0x01, 0 },
        { 0x0e, 0x16, 0x01, 0 }
    };
    uchar in_pkt[MSG_SIZE];
    int modes = (IS_K95(kb) ? HWMODE_K95 : HWMODE_K70);
    for(int i = 0; i <= modes; i++){
        data_pkt[0][3] = i;
        if(!usbrecv(kb, data_pkt[0], in_pkt)){
            free(hw);
            return -1;
        }
        memcpy(hw->id + i, in_pkt + 4, sizeof(usbid));
    }
    // Ask for profile name
    if(!usbrecv(kb, data_pkt[1], in_pkt)){
        free(hw);
        return -1;
    }
    memcpy(hw->name[0], in_pkt + 4, PR_NAME_LEN * 2);
    // Load modes
    for(int i = 0; i < modes; i++){
        if(hwloadmode(kb, hw, i)){
            free(hw);
            return -1;
        }
    }
    // Make the profile active (if requested)
    if(apply)
        hwtonative(kb->profile, hw, modes);
    // Free the existing profile (if any)
    free(kb->hw);
    kb->hw = hw;
    DELAY_LONG(kb);
    return 0;
}
```

Here is the call graph for this function:



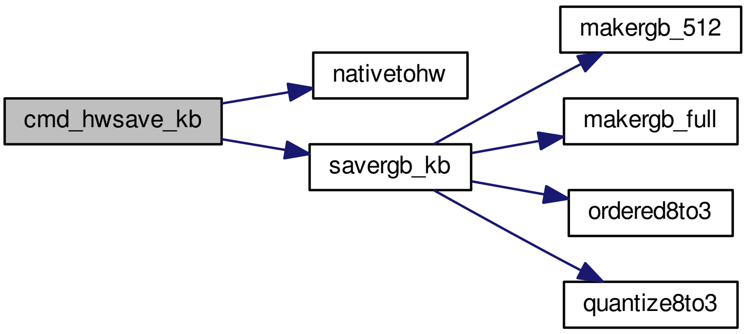
5.34.1.2 int cmd_hwsave_kb (usbdevice ∗ *kb,* usbmode ∗ *dummy1,* int *dummy2,* int *dummy3,* const char ∗ *dummy4*)

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.

```
                                                                                                    {
    DELAY_LONG(kb);
    hwprofile* hw = kb->hw;
    if(!hw)
        hw = kb->hw = calloc(1, sizeof(hwprofile));
    int modes = (IS_K95(kb) ? HWMODE_K95 : HWMODE_K70);
    nativetohw(kb->profile, hw, modes);
    // Save the profile and mode names
    uchar data_pkt[2][MSG_SIZE] = {
        { 0x07, 0x16, 0x01, 0 },
        { 0x07, 0x15, 0x01, 0 },
    };
    // Save the mode names
    for(int i = 0; i <= modes; i++){
        data_pkt[0][3] = i;
        memcpy(data_pkt[0] + 4, hw->name[i], MD_NAME_LEN * 2);
        if(!usbsend(kb, data_pkt[0], 1))
            return -1;
    }
    // Save the IDs
    for(int i = 0; i <= modes; i++){
        data_pkt[1][3] = i;
        memcpy(data_pkt[1] + 4, hw->id + i, sizeof(usbid));
        if(!usbsend(kb, data_pkt[1], 1))
            return -1;
    }
    // Save the RGB data
    for(int i = 0; i < modes; i++){
        if(savergb_kb(kb, hw->light + i, i))
            return -1;
    }
    DELAY_LONG(kb);
    return 0;
}
```

Here is the call graph for this function:



5.34.1.3 static int hwloadmode (usbdevice * *kb,* hwprofile * *hw,* int *mode*) [static]

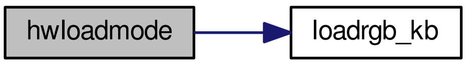
Definition at line 5 of file profile_keyboard.c.

References hwprofile::light, loadrgb_kb(), MD_NAME_LEN, MSG_SIZE, hwprofile::name, and usbrecv.

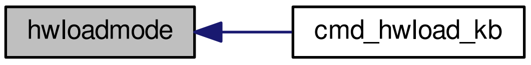
Referenced by cmd_hwload_kb().

```
                                                              {
    // Ask for mode's name
    uchar data_pkt[MSG_SIZE] = { 0x0e, 0x16, 0x01, mode + 1, 0 };
    uchar in_pkt[MSG_SIZE];
    if(!usbrecv(kb, data_pkt, in_pkt))
        return -1;
    memcpy(hw->name[mode + 1], in_pkt + 4, MD_NAME_LEN * 2);
    // Load the RGB setting
    return loadrgb_kb(kb, hw->light + mode, mode);
}
```

Here is the call graph for this function:



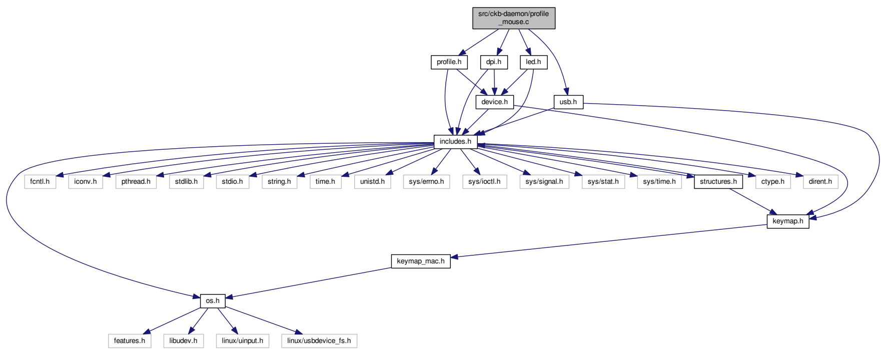
Here is the caller graph for this function:



5.35 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)

5.35.1 Function Documentation

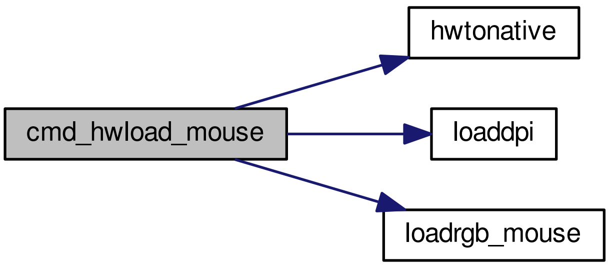
5.35.1.1 int cmd_hwload_mouse (**usbdevice** ∗ *kb,* **usbmode** ∗ *dummy1,* int *dummy2,* int *apply,* const char ∗ *dummy3*)

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.

```
{
    DELAY_LONG(kb);
    hwprofile* hw = calloc(1, sizeof(hwprofile));
    // Ask for profile and mode IDs
     uchar data_pkt[2][MSG_SIZE] = {
         { 0x0e, 0x15, 0x01, 0 },
         { 0x0e, 0x16, 0x01, 0 }
     };
     uchar in_pkt[MSG_SIZE];
     for(int i = 0; i <= 1; i++){
         data_pkt[0][3] = i;
         if(!usbrecv(kb, data_pkt[0], in_pkt)){
             free(hw);
             return -1;
         }
         memcpy(hw->id + i, in_pkt + 4, sizeof(usbid));
     }
     // Ask for profile and mode names
     for(int i = 0; i <= 1; i++){
         data_pkt[1][3] = i;
         if(!usbrecv(kb, data_pkt[1],in_pkt)){
             free(hw);
             return -1;
         }
         memcpy(hw->name[i], in_pkt + 4, PR_NAME_LEN * 2);
     }

     // Load the RGB and DPI settings
     if(loadrgb_mouse(kb, hw->light, 0)
             || loaddpi(kb, hw->dpi, hw->light)){
         free(hw);
         return -1;
     }

     // Make the profile active (if requested)
     if(apply)
         hwtonative(kb->profile, hw, 1);
     // Free the existing profile (if any)
     free(kb->hw);
     kb->hw = hw;
     DELAY_LONG(kb);
     return 0;
}
```

Here is the call graph for this function:



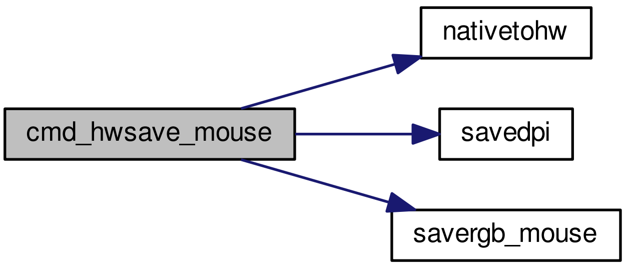
5.35.1.2 int cmd_hwsave_mouse (**usbdevice** ∗ *kb,* **usbmode** ∗ *dummy1,* int *dummy2,* int *dummy3,* const char ∗ *dummy4*)

Definition at line 50 of file profile_mouse.c.

References DELAY_LONG, hwprofile::dpi, usbdevice::hw, hwprofile::id, hwprofile::light, MD_NAME_LEN, MSG_S-IZE, hwprofile::name, nativetohw(), usbdevice::profile, savedpi(), savergb_mouse(), and usbsend.

```
                                                                                    {
    DELAY_LONG(kb);
    hwprofile* hw = kb->hw;
    if(!hw)
        hw = kb->hw = calloc(1, sizeof(hwprofile));
    nativetohw(kb->profile, hw, 1);
    // Save the profile and mode names
    uchar data_pkt[2][MSG_SIZE] = {
        { 0x07, 0x16, 0x01, 0 },
        { 0x07, 0x15, 0x01, 0 },
    };
    for(int i = 0; i <= 1; i++){
        data_pkt[0][3] = i;
        memcpy(data_pkt[0] + 4, hw->name[i], MD_NAME_LEN * 2);
        if(!usbsend(kb, data_pkt[0], 1))
            return -1;
    }
    // Save the IDs
    for(int i = 0; i <= 1; i++){
        data_pkt[1][3] = i;
        memcpy(data_pkt[1] + 4, hw->id + i, sizeof(usbid));
        if(!usbsend(kb, data_pkt[1], 1))
            return -1;
    }
    // Save the RGB data for the non-DPI zones
    if(savergb_mouse(kb, hw->light, 0))
        return -1;
    // Save the DPI data (also saves RGB for those states)
    if(savedpi(kb, hw->dpi, hw->light))
        return -1;
    DELAY_LONG(kb);
    return 0;
}
```

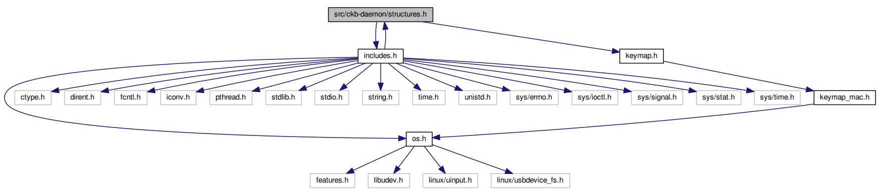
Here is the call graph for this function:



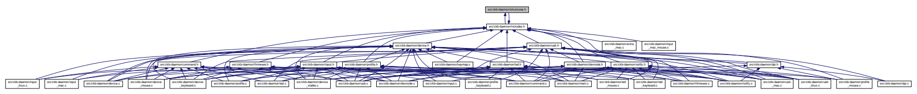
5.36 src/ckb-daemon/structures.h File Reference

```
#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 34
- #define SERIAL_LEN 34
- #define MSG_SIZE 64
- #define IFACE_MAX 4

Variables

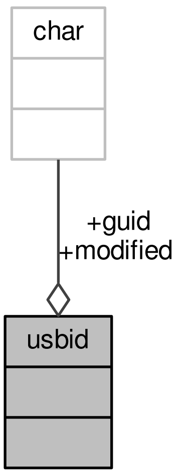
- const union devcmd vtable_keyboard
- const union devcmd vtable_keyboard_nonrgb
- const union devcmd vtable_mouse

5.36.1 Data Structure Documentation

5.36.1.1 struct usbid

Definition at line 8 of file structures.h.

Collaboration diagram for usbid:



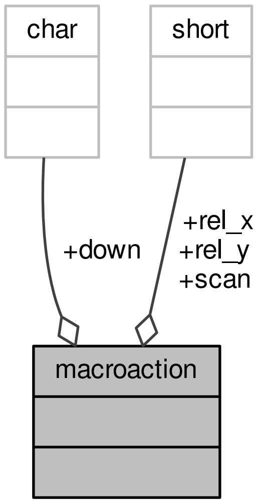
Data Fields

char	guid[16]	
char	modified[4]	

5.36.1.2 struct macroaction

Definition at line 27 of file structures.h.

Collaboration diagram for macroaction:



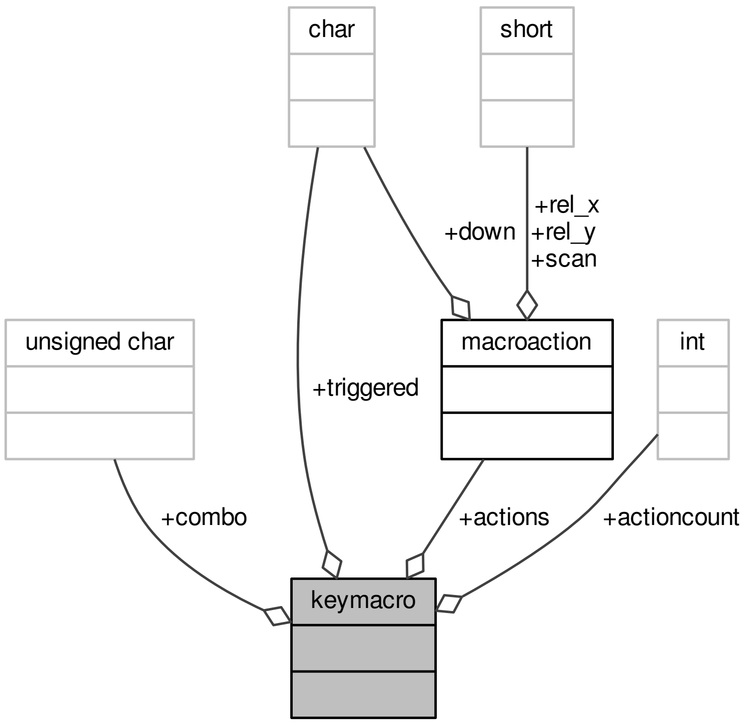
Data Fields

char	down	
short	rel_x	
short	rel_y	
short	scan	

5.36.1.3 struct keymacro

Definition at line 34 of file structures.h.

Collaboration diagram for keymacro:



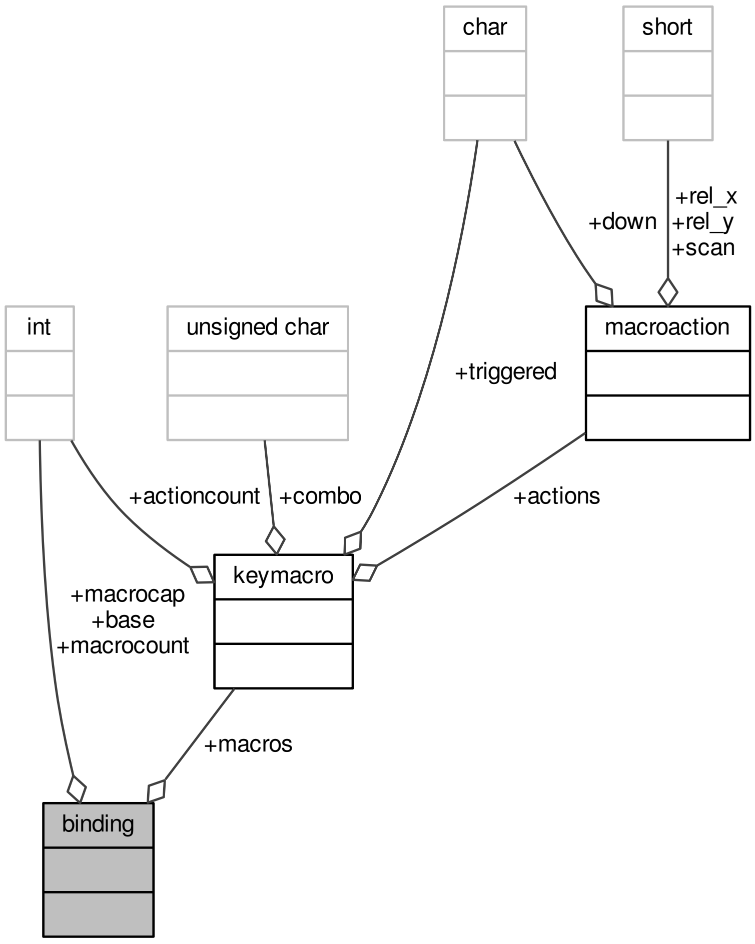
Data Fields

int	actioncount	
macroaction ∗	actions	
uchar	combo[((((152+3+12)+25)+7)/8)]	
char	triggered	

5.36.1.4 struct binding

Definition at line 42 of file structures.h.

Collaboration diagram for binding:



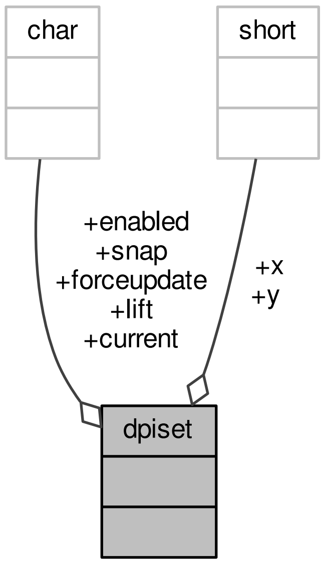
Data Fields

int	base[((152+3+12)+25)]	
int	macrocap	
int	macrocount	
keymacro ∗	macros	

5.36.1.5 struct dpiset

Definition at line 56 of file structures.h.

Collaboration diagram for dpiset:



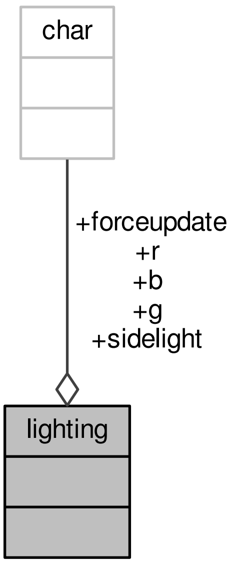
Data Fields

uchar	current	
uchar	enabled	
uchar	forceupdate	
uchar	lift	
uchar	snap	
ushort	x[6]	
ushort	y[6]	

5.36.1.6 struct lighting

Definition at line 72 of file structures.h.

Collaboration diagram for lighting:



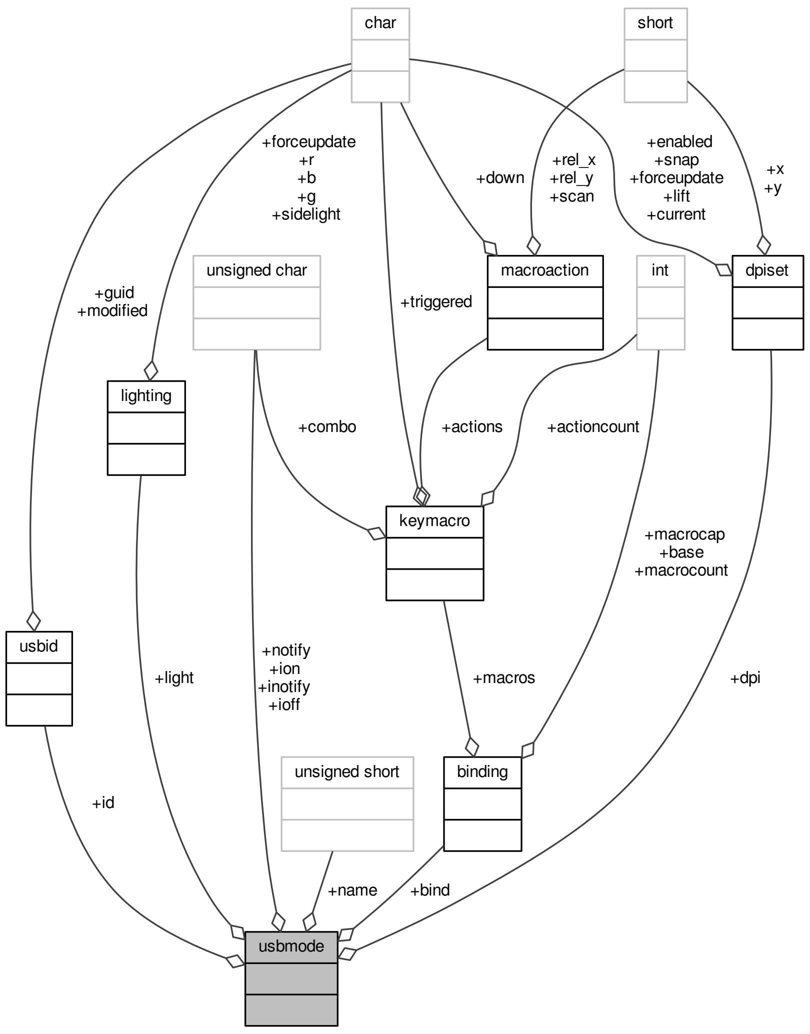
Data Fields

uchar	b[152+11]	
uchar	forceupdate	
uchar	g[152+11]	
uchar	r[152+11]	
uchar	sidelight	

5.36.1.7 struct usbmode

Definition at line 82 of file structures.h.

Collaboration diagram for usbmode:



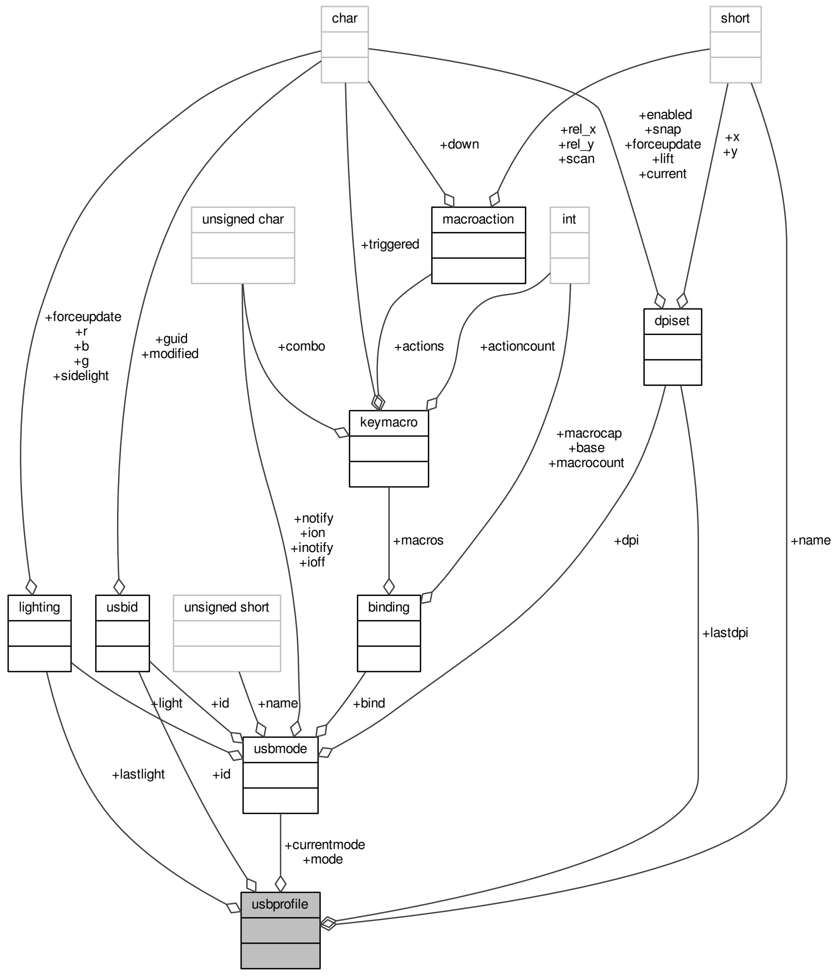
Data Fields

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)]	

5.36.1.8 struct usbprofile

Definition at line 100 of file structures.h.

Collaboration diagram for usbprofile:



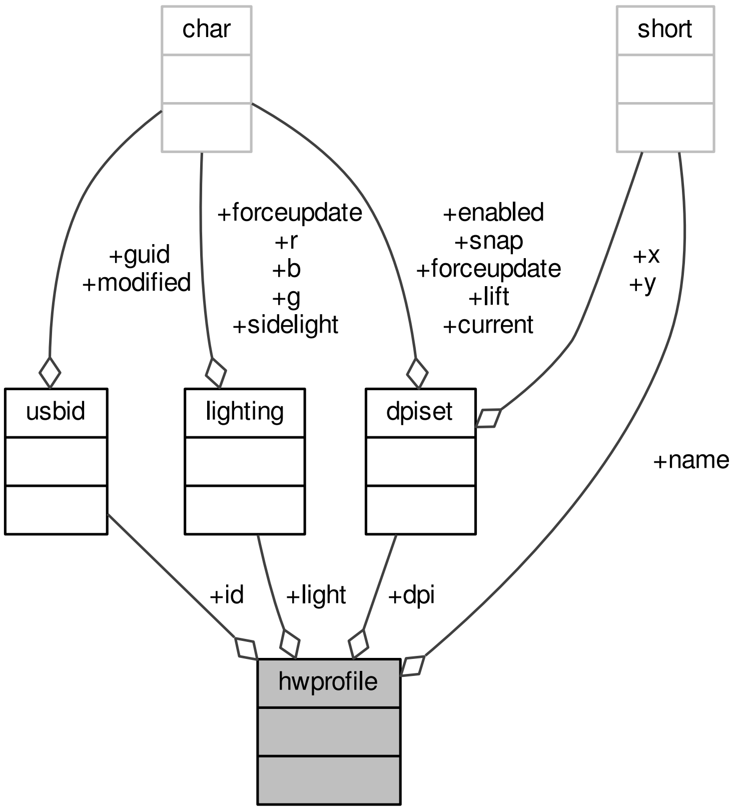
Data Fields

usbmode ∗	currentmode	
usbid	id	
dpiset	lastdpi	
lighting	lastlight	
usbmode	mode[6]	
ushort	name[16]	

5.36.1.9 struct hwprofile

Definition at line 117 of file structures.h.

Collaboration diagram for hwprofile:



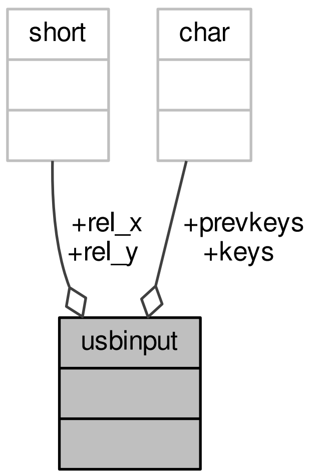
Data Fields

dpiset	dpi[3]	
usbid	id[3+1]	
lighting	light[3]	
ushort	name[3+1][16]	

5.36.1.10 struct usbinput

Definition at line 128 of file structures.h.

Collaboration diagram for usbinput:



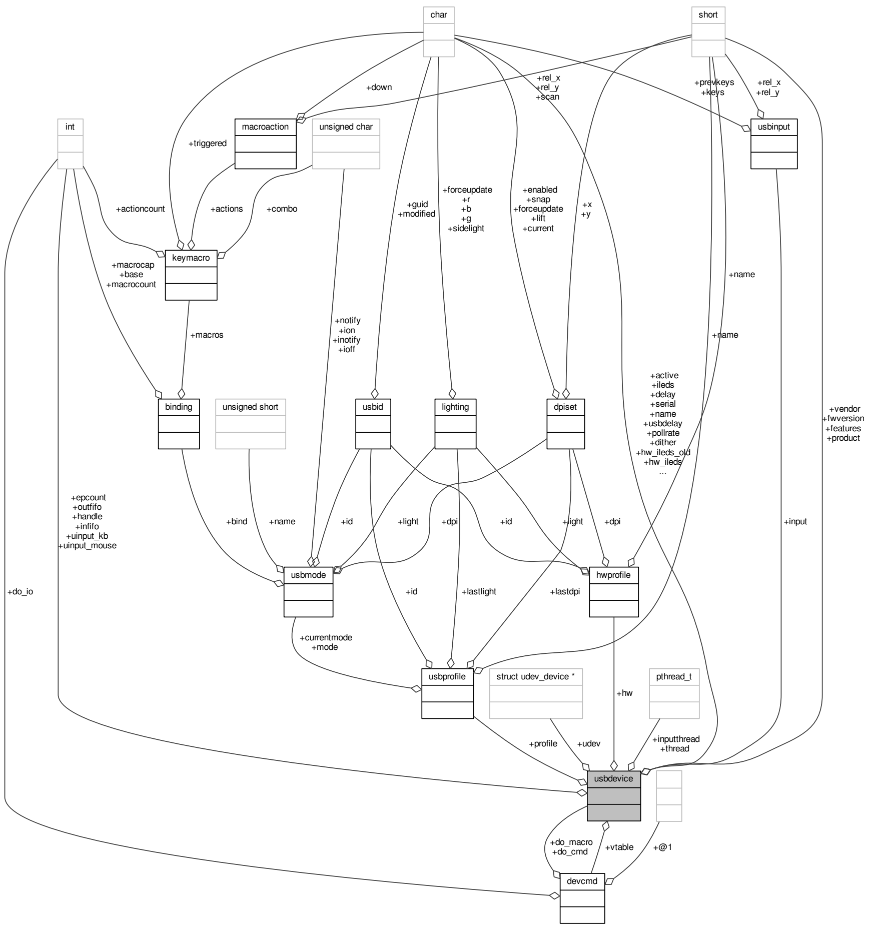
Data Fields

uchar	keys[((((152+3+12)+25)+7)/8)]
uchar	prevkeys[((((152+3+12)+25)+7)/8)]
short	rel_x
short	rel_y

5.36.1.11 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[34+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	

5.36.2 Macro Definition Documentation

5.36.2.1 #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().

5.36.2.2 #define DPI_COUNT 6

Definition at line 53 of file structures.h.

Referenced by cmd_dpi(), cmd_dpisel(), loaddpi(), printdpi(), savedpi(), and updatedpi().

5.36.2.3 #define FEAT_ADJRATE 0x008

Definition at line 138 of file structures.h.

Referenced by _mkdevpath(), _setupusb(), and _start_dev().

5.36.2.4 #define FEAT_ANSI 0x200

Definition at line 145 of file structures.h.

Referenced by readcmd().

5.36.2.5 #define FEAT_BIND 0x010

Definition at line 139 of file structures.h.

Referenced by _mkdevpath(), main(), and readcmd().

5.36.2.6 #define FEAT_COMMON (FEAT_BIND | FEAT_NOTIFY | FEAT_FWVERSION | FEAT_MOUSEACCEL | FEAT_HWLOAD)

Definition at line 150 of file structures.h.

5.36.2.7 #define FEAT_FWUPDATE 0x080

Definition at line 142 of file structures.h.

Referenced by _mkdevpath(), _start_dev(), and cmd_fwupdate().

5.36.2.8 #define FEAT_FWVERSION 0x040

Definition at line 141 of file structures.h.

Referenced by _mkdevpath(), and _start_dev().

5.36.2.9 #define FEAT_HWLOAD 0x100

Definition at line 143 of file structures.h.

Referenced by _start_dev().

5.36.2.10 #define FEAT_ISO 0x400

Definition at line 146 of file structures.h.

Referenced by readcmd().

5.36.2.11 #define FEAT_LMASK (FEAT_ANSI | FEAT_ISO)

Definition at line 153 of file structures.h.

Referenced by readcmd().

5.36.2.12 #define FEAT_MONOCHROME 0x002

Definition at line 136 of file structures.h.

Referenced by _mkdevpath(), and _setupusb().

5.36.2.13 #define FEAT_MOUSEACCEL 0x800

Definition at line 147 of file structures.h.

Referenced by main(), and readcmd().

5.36.2.14 #define FEAT_NOTIFY 0x020

Definition at line 140 of file structures.h.

Referenced by _mkdevpath(), main(), and readcmd().

5.36.2.15 #define FEAT_POLLRATE 0x004

Definition at line 137 of file structures.h.

Referenced by _mkdevpath(), _start_dev(), and getfwversion().

5.36.2.16 #define FEAT_RGB 0x001

Definition at line 135 of file structures.h.

Referenced by _mkdevpath(), _start_dev(), os_setupusb(), revertusb(), and usbunclaim().

5.36.2.17 #define FEAT_STD_NRGB (FEAT_COMMON)

Definition at line 152 of file structures.h.

Referenced by _setupusb().

5.36.2.18 #define FEAT_STD_RGB (FEAT_COMMON | FEAT_RGB | FEAT_POLLRATE | FEAT_FWUPDATE)

Definition at line 151 of file structures.h.

Referenced by _setupusb().

5.36.2.19 #define HAS_ANY_FEATURE(*kb, feat*) (!!((kb)->features & (feat)))

Definition at line 157 of file structures.h.

5.36.2.20 #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 usbun-claim().

5.36.2.21 #define HWMODE_K70 1

Definition at line 114 of file structures.h.

Referenced by cmd_hwload_kb(), and cmd_hwsave_kb().

5.36.2.22 #define HWMODE_K95 3

Definition at line 115 of file structures.h.

Referenced by cmd_hwload_kb(), and cmd_hwsave_kb().

5.36.2.23 #define HWMODE_MAX 3

Definition at line 116 of file structures.h.

5.36.2.24 #define I_CAPS 2

Definition at line 20 of file structures.h.

Referenced by _cmd_get(), iselect(), nprintind(), and updateindicators_kb().

5.36.2.25 #define I_NUM 1

Definition at line 19 of file structures.h.

Referenced by _cmd_get(), iselect(), nprintind(), and updateindicators_kb().

5.36.2.26 #define I_SCROLL 4

Definition at line 21 of file structures.h.

Referenced by _cmd_get(), iselect(), nprintind(), and updateindicators_kb().

5.36.2.27 #define IFACE_MAX 4

Definition at line 176 of file structures.h.

5.36.2.28 #define KB_NAME_LEN 34

Definition at line 173 of file structures.h.

Referenced by _setupusb(), and os_setupusb().

5.36.2.29 #define LIFT_MAX 5

Definition at line 55 of file structures.h.

Referenced by cmd_lift(), and loaddpi().

5.36.2.30 #define LIFT_MIN 1

Definition at line 54 of file structures.h.

Referenced by cmd_lift(), and loaddpi().

5.36.2.31 #define MACRO_MAX 1024

Definition at line 50 of file structures.h.

Referenced by _cmd_macro().

5.36.2.32 #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().

5.36.2.33 #define MODE_COUNT 6

Definition at line 99 of file structures.h.

Referenced by _freeprofile(), allocprofile(), and readcmd().

5.36.2.34 #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().

5.36.2.35 #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().

5.36.2.36 #define OUTFIFO_MAX 10

Definition at line 24 of file structures.h.

Referenced by _mknotifynode(), _rmnotifynode(), inputupdate_keys(), nprintf(), readcmd(), rmdevpath(), and updateindicators_kb().

5.36.2.37 #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().

5.36.2.38 #define SCROLL_ACCELERATED 0

Definition at line 163 of file structures.h.

Referenced by readcmd().

5.36.2.39 #define SCROLL_MAX 10

Definition at line 165 of file structures.h.

Referenced by readcmd().

5.36.2.40 #define SCROLL_MIN 1

Definition at line 164 of file structures.h.

Referenced by readcmd().

5.36.2.41 #define SERIAL_LEN 34

Definition at line 174 of file structures.h.

Referenced by _setupusb(), and os_setupusb().

5.36.2.42 #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().

5.36.3 Variable Documentation

5.36.3.1 const union devcmd vtable_keyboard

Definition at line 28 of file device_vtable.c.

Referenced by get_vtable().

5.36.3.2 const union devcmd vtable_keyboard_nonrgb

Definition at line 75 of file device_vtable.c.

Referenced by get_vtable().

5.36.3.3 const union **devcmd** vtable_mouse

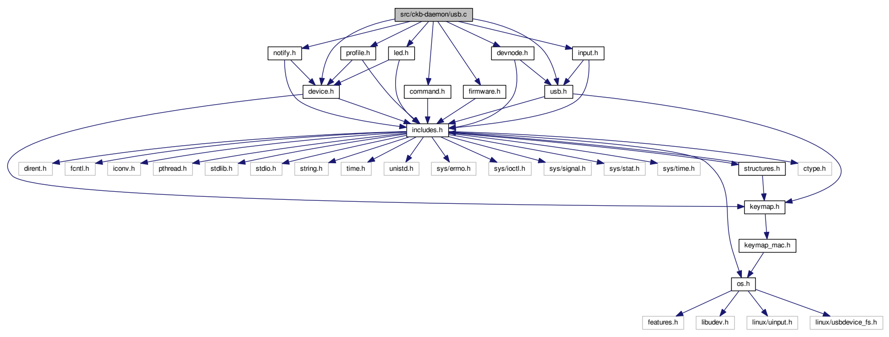
Definition at line 122 of file device_vtable.c.

Referenced by get_vtable().

5.37 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)

    brief .
- const char ∗ product_str (short product)

    brief .
- static const devcmd ∗ get_vtable (short vendor, short product)

    brief .
- static void ∗ devmain (usbdevice ∗kb)

    brief .
- static void ∗ _setupusb (void ∗context)

    brief .
- 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

  brief .
- volatile int reset_stop = 0

  brief .
- int features_mask = -1

  brief .
- int hwload_mode

5.37.1 Function Documentation

5.37.1.1 int _resetusb (usbdevice ∗ *kb,* const char ∗ *file,* int *line*)

_resetusb Reset a USB device.

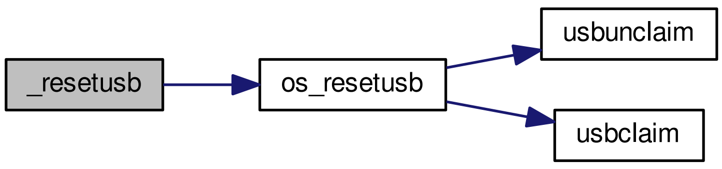
First reset the device via os_resetusb() after a long delay (it may send something to the host). If this worked (retval == 0), give the device another long delay Then perform the initialization via the device specific start() function entry in kb->vtable and if this is successful also, return the result of the device depenten updatergb() with force=true.

Definition at line 426 of file usb.c.

References usbdevice::active, DELAY_LONG, os_resetusb(), and usbdevice::vtable.

```
                                                         {
    // Perform a USB reset
    DELAY_LONG(kb);
    int res = os_resetusb(kb, file, line);
    if(res)
        return res;
    DELAY_LONG(kb);
    // Re-initialize the device
    if(kb->vtable->start(kb, kb->active) != 0)
        return -1;
    if(kb->vtable->updatergb(kb, 1) != 0)
        return -1;
    return 0;
}
```

Here is the call graph for this function:



5.37.1.2 static void∗ _setupusb (void ∗ *context*) [static]

_setupusb A horrible function for setting up an usb device

Parameters

context	As _setupusb() is called as a new thread, the kb∗ is transferred as void∗

Returns

a pthread_t∗ 0, here casted as void∗. Retval is always null

The basic structure of the function is somewhat habituated. It is more like an assembler routine than a structured program. This is not really bad, but just getting used to.

After every action, which can be practically fault-prone, the routine goes into the same error handling: It goes via goto to one of two exit labels. The difference is whether or not an unlock has to be performed on the imutex variable. In both cases, closeusb() is called, then an unlock is performed on the dmutex.

The only case where this error handling is not performed is the correct return of the call to devmain(). Here simply the return value of devmain() is passed to the caller.

In either case, the routine terminates with a void∗ 0 because either devmain() has returned constant null or the routine itself returns zero.

The basic idea of this routine is the following:

First some initialization of kb standard structured and local vars is done.

- **kb** is set to the pointer given from start environment
- local vars **vendor** and **product** are set to the values from the corresponding fields of kb
- local var **vt and** the **kb->vtable** are both set to the retval of *get_vtable()*
- **kb->features** are set depending on the type of hardware connected:
  - set either to standard non rgb (all common flags like binding, notify, FW, hardware-loading etc) or in case of RGB-device set to standard + RGB, pollrate-change and fw-update
  - exclude all features which are disabled via feature_mask (set by daemon CLI parameters)
  - if it is a mouse, add adjust-rate
  - if it is a monochrome device, set the flag for RGB-protocol, but single color
- the standard delay time is initialized in kb->usbdelay
- A fixed 100ms wait is the start. **Although the DELAY_LONG macro is given a parameter, it is ignored. Occasionally refactor it.**
- The first relevant point is the operating system-specific opening of the interface in os_setupusb(). As a result, some parameters should be set in kb (name, serial, fwversion, epcount = number of usb endpoints), and all endpoints should be claimed with usbclaim(). Claiming is the only point where os_setupusb() can produce an error (-1, otherwise 0).
- The following two statements deal with possible errors when setting the kb values in the current routine: If the version or the name was not read correctly, they are set to default values:
  - serial is set to "<vendor>: <product> -NoID"
  - the name is set to "<vendor> <product>".
- Then the user level input subsystem is activated via os_openinput(). There are two file descriptors, one for the mouse and one for the keyboard. **As mentioned in structures.h, not the just opened FD numbers are stored under kb->uinput_kb or kb->uinput_mouse, but the values increased by 1!** The reason is, if the open fails or not open has been done until now, that struct member is set to 0, not to -1 or other negative value. So all usage of this kb->handle must be something like "`kb->handle - 1`", as you can find it in the code.
- The next action is to create a separate thread, which gets as parameter kb and starts with os_inputmain(). The thread is immediately detached so that it can return its resource completely independently if it should terminate.

- The same happens with os_setupindicators(), which initially initializes all LED variables in kb to off and then starts the _ledthread() thread with kb as parameter and then detaches it. Here again only the generation of the thread can fail.
- Via an entry in the vable (allocprofile, identical for all three vtable types), allocprofile() is called in profile.c. With a valid parameter kb, a usbprofile structure is allocated and stored as a kb->profile. Then initmode() is called for each of the initializable modes (MODE_COUNT, currently 6). This procedure creates the memory space for the mode information, initializes the range to 0, and then sets the light.forceupdate and dpi.forceupdate to true. This forces an update later in the initialization of the device.

  The first mode is set as the current mode and two force flags are set (this seems to be mode-intersecting flags for light and update).

  Warning

  > There is no error handling for the allocprofile() and initmode() procedures. However, since they allocate storage areas, the subsequent assignments and initializations can run in a SEGV.

- Not completely understandable is why now via the vtable the function updateindicators() is called. But this actually happens in the just started thread _ledthread(). Either the initialization is wrong und must done here with force or the overview is lost, what happens when...

  Regardless: For a mouse nothing happens here, for a keyboard updateindicators_kb() is called via the entry in kb->vtable. The first parameter is kb again, the second is constant 1 (means force = true). This causes the LED status to be sent after a 5ms delay via os_sendindicators() (ioctl with a `usbdevfs_ctrltransfer`).

  The notification is sent to all currently open notification channels then.

  Setupindicators() and with it updateindicators_kb() can fail.

- From this point - if an error is detected - the error label is addressed by goto statement, which first performs an unlock on the imutex. This is interesting because the next statement is exactly this: An unlock on the imutex.
- Via vtable the *kb->start()* function is called next. This is the same for a mouse and an RGB keyboard: start_dev(), for a non RGB keyboard it is start_kb_nrgb().

  First parameter is as always kb, second is 0 (makeactive = false).

  - In start_kb_nrgb() set the keyboard into a so-called software mode (NK95_HWOFF) via ioctl with `usbdevfs_ctrltransfer` in function _nk95cmd(), which will in turn is called via macro nk95cmd() via start_kb_nrgb().

    Then two dummy values (active and pollrate) are set in the kb structure and ready.
  - start_dev() does a bit more - because this function is for both mouse and keyboard. start_dev() calls - after setting an extended timeout parameter - _start_dev(). Both are located in device.c.
  - First, _start_dev() attempts to determine the firmware version of the device, but only if two conditions are met: hwload-mode is not null (then hw-loading is disabled) and the device has the FEAT_HWLOAD feature. Then the firmware and the poll rate are fetched via getfwversion().

    If hwload_mode is set to "load only once" (==1), then the HWLOAD feature is masked, so that no further reading can take place.
  - Now check if device needs a firmware update. If so, set it up and leave the function without error.
  - Else load the hardware profile from device if the hw-pointer is not set and hw-loading is possible and allowed.

    Return error if mode == 2 (load always) and loading got an error. Else mask the HWLOAD feature, because hwload must be 1 and the error couold be a repeated hw-reading.

    Puh, that is real Horror code. It seems to be not faulty, but completely unreadable.
  - Finally, the second parameter of _startdev() is used to check whether the device is to be activated. Depending on the parameter, the active or the idle-member in the correspondig vtable is called. These are device-dependent again:

Device	active	idle
RGB Keyboard	cmd_active_kb() means: start the device with a lot of kb-specific initializers (software controlled mode)	cmd_idle_kb() set the device with a lot of kb-specific initializers into the hardware controlled mode)
non RGB Keyboard	cmd_io_none() means: Do nothing	cmd_io_none() means: Do nothing
Mouse	cmd_active_mouse() similar to cmd_active_kb()	cmd_idle_mouse similar to cmd_idle_kb()

- If either *start()* succeeded or the next following usb_tryreset(), it goes on, otherwise again a hard abort occurs.
- Next, go to mkdevpath(). After securing the EUID (effective UID) especially for macOS, work starts really in _mkdevpath(). Create - no matter how many devices were registered - either the ckb0/ files **version**, **pid** and **connected** or the **cmd** command fifo, the first notification fifo **notify0**, **model** and **serial** as well as the **features** of the device and the **pollrate**.
- If all this is done and no error has occurred, a debug info is printed ("Setup finished for ckbx") updateconnected() writes the new device into the text file under ckb0/ and devmain() is called.

devmain()'s return value is returned by _setupusb() when we terminate.

- The remaining code lines are the two exit labels as described above

Definition at line 214 of file usb.c.

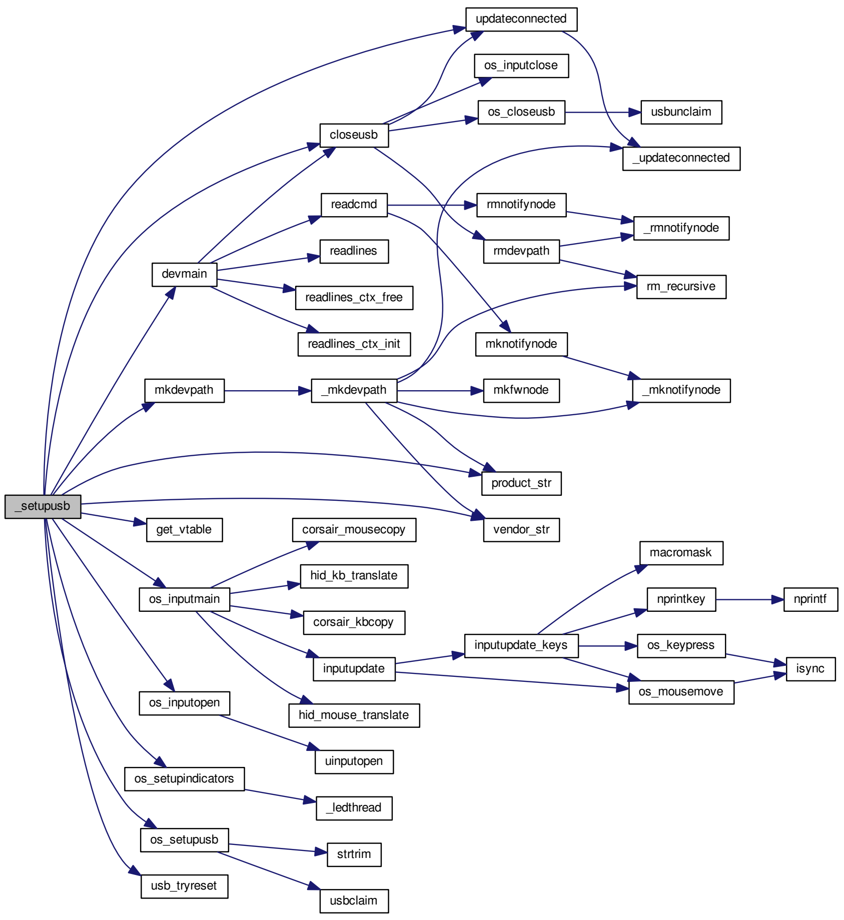
References ckb_info, closeusb(), DELAY_LONG, devmain(), devpath, dmutex, FEAT_ADJRATE, FEAT_MONOCHROME, FEAT_STD_NRGB, FEAT_STD_RGB, usbdevice::features, features_mask, get_vtable(), imutex, INDEX_OF, usbdevice::inputthread, IS_MONOCHROME, IS_MOUSE, IS_RGB, KB_NAME_LEN, keyboard, mkdevpath(), usbdevice::name, 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().

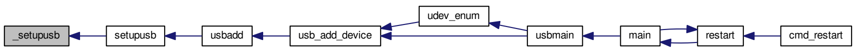
```
                                {
  usbdevice* kb = context;
  // Set standard fields
  short vendor = kb->vendor, product = kb->product;
  const devcmd* vt = kb->vtable = get_vtable(vendor, product);
  kb->features = (IS_RGB(vendor, product) ? FEAT_STD_RGB :
FEAT_STD_NRGB) & features_mask;
  if(IS_MOUSE(vendor, product)) kb->features |= FEAT_ADJRATE;
  if(IS_MONOCHROME(vendor, product)) kb->features |=
FEAT_MONOCHROME;
  kb->usbdelay = USB_DELAY_DEFAULT;

  // Perform OS-specific setup
  DELAY_LONG(kb);

  if(os_setupusb(kb))
      goto fail;

  // Make up a device name and serial if they weren't assigned
  if(!kb->serial[0])
      snprintf(kb->serial, SERIAL_LEN, "%04x:%04x-NoID", kb->
vendor, kb->product);
  if(!kb->name[0])
      snprintf(kb->name, KB_NAME_LEN, "%s %s", vendor_str(kb->
vendor), product_str(kb->product));

  // Set up an input device for key events
  if(os_inputopen(kb))
      goto fail;
  if(pthread_create(&kb->inputthread, 0, os_inputmain, kb))
      goto fail;
  pthread_detach(kb->inputthread);
  if(os_setupindicators(kb))
      goto fail;

  // Set up device
  vt->allocprofile(kb);
```

```
    vt->updateindicators(kb, 1);
    pthread_mutex_unlock(imutex(kb));
    if(vt->start(kb, 0) && usb_tryreset(kb))
        goto fail_noinput;
    // Make /dev path
    if(mkdevpath(kb))
        goto fail_noinput;
    // Finished. Enter main loop
    int index = INDEX_OF(kb, keyboard);
    ckb_info("Setup finished for %s%d\n", devpath, index);
    updateconnected();
    return devmain(kb);
    fail:
    pthread_mutex_unlock(imutex(kb));
    fail_noinput:
    closeusb(kb);
    pthread_mutex_unlock(dmutex(kb));
    return 0;
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.37.1.3 int _usbrecv (usbdevice * *kb,* const uchar * *out_msg,* uchar * *in_msg,* const char * *file,* int *line*)

_usbrecv Request data from a USB device by first sending an output packet and then reading the response.

To fully understand this, you need to know about usb: All control is at the usb host (the CPU). If the device wants to communicate something to the host, it must wait for the host to ask. The usb protocol defines the cycles and periods in which actions are to be taken.

So in order to receive a data packet from the device, the host must first send a send request.

This is done by _usbrecv() in the first block by sending the MSG_SIZE large data block from **out_msg** via os_usbsend() as it is a machine depending implementation. The usb target device is as always determined over kb.

For os_usbsend() to know that it is a receive request, the **is_recv** parameter is set to true (1). With this, os_usbsend () generates a control package for the hardware, not a data packet.

If sending of the control package is not successful, a maximum of 5 times the transmission is repeated (including the first attempt). If a non-cancelable error is signaled or the drive is stopped via reset_stop, _usbrecv() immediately returns 0.

After this, the function waits for the requested response from the device using os_usbrecv ().

os_usbrecv() returns 0, -1 or something else.

Zero signals a serious error which is not treatable and _usbrecv() also returns 0.

-1 means that it is a treatable error - a timeout for example - and therefore the next transfer attempt is started after a long pause (DELAY_LONG) if not reset_stop or the wrong hwload_mode require a termination with a return value of 0.

After 5 attempts, _usbrecv () returns and returns 0 as well as an error message.

When data is received, the number of received bytes is returned. This should always be MSG_SIZE, but os_usbrecv() can also return less. It should not be more, because then there would be an unhandled buffer overflow, but it could be less. This would be signaled in os_usbrecv () with a message.

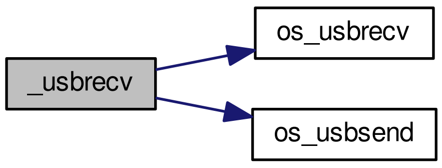
The buffers behind **out_msg** and **in_msg** are MSG_SIZE at least (currently 64 Bytes). More is ok but useless, less brings unpredictable behavior.

Definition at line 599 of file usb.c.

References ckb_err_fn, DELAY_LONG, DELAY_MEDIUM, DELAY_SHORT, hwload_mode, os_usbrecv(), os_usbsend(), and reset_stop.

```
                                                                                       {
    // Try a maximum of 3 times
    for(int try = 0; try < 5; try++){
        // Send the output message
        DELAY_SHORT(kb);
        int res = os_usbsend(kb, out_msg, 1, file, line);
        if(res == 0)
            return 0;
        else if(res == -1){
            // Retry on temporary failure
            if(reset_stop)
                return 0;
            DELAY_LONG(kb);
            continue;
        }
        // Wait for the response
        DELAY_MEDIUM(kb);
        res = os_usbrecv(kb, in_msg, file, line);
        if(res == 0)
            return 0;
        else if(res != -1)
            return res;
        if(reset_stop || hwload_mode != 2)
            return 0;
        DELAY_LONG(kb);
    }
    // Give up
    ckb_err_fn("Too many send/recv failures. Dropping.\n", file, line);
    return 0;
}
```

Here is the call graph for this function:



5.37.1.4 int _usbsend (**usbdevice** ∗ *kb,* const **uchar** ∗ *messages,* int *count,* const char ∗ *file,* int *line*)

_usbsend send a logical message completely to the given device

Todo A lot of different conditions are combined in this code. Don't think, it is good in every combination...

The main task of _usbsend () is to transfer the complete logical message from the buffer beginning with *messages* to **count** ∗ **MSG_SIZE**.

According to usb 2.0 specification, a USB transmits a maximum of 64 byte user data packets. For the transmission of longer messages we need a segmentation. And that is exactly what happens here.

The message is given one by one to os_usbsend() in MSG_SIZE (= 64) byte large bites.

Attention

This means that the buffer given as argument must be n ∗ MSG_SIZE Byte long.

An essential constant parameter which is relevant for os_usbsend() only is is_recv = 0, which means sending.

Now it gets a little complicated again:

- If os_usbsend() returns 0, only zero bytes could be sent in one of the packets, or it was an error (-1 from the systemcall), but not a timeout. How many Bytes were sent in total from earlier calls does not seem to matter, _usbsend() returns a total of 0.
- Returns os_usbsend() -1, first check if **reset_stop** is set globally or (incomprehensible) hwload_mode is not set to "always". In either case, _usbsend() returns 0, otherwise it is assumed to be a temporary transfer error and it simply retransmits the physical packet after a long delay.
- If the return value of os_usbsend() was neither 0 nor -1, it specifies the numer of bytes transferred.

  Here is an information hiding conflict with os_usbsend() (at least in the Linux version):

  If os_usbsend() can not transfer the entire packet, errors are thrown and the number of bytes sent is returned. _usbsend() interprets this as well and remembers the total number of bytes transferred in the local variable **total_sent**. Subsequently, however, transmission is continued with the next complete MSG_SIZE block and not with the first of the possibly missing bytes.

  Todo Check whether this is the same in the macOS variant. It is not dramatic, but if errors occur, it can certainly irritate the devices completely if they receive incomplete data streams. Do we have errors with the messages "Wrote YY bytes (expected 64)" in the system logs? If not, we do not need to look any further.

When the last packet is transferred, _usbsend() returns the effectively counted set of bytes (from **total_sent**). This at least gives the caller the opportunity to check whether something has been lost in the middle.

A bit strange is the structure of the program: Handling the **count** MSG_SIZE blocks to be transferred is done in the outer for (...) loop. Repeating the transfer with a treatable error is managed by the inner while(1) loop.

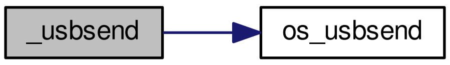
This must be considered when reading the code; The "break" on successful block transfer leaves the inner while, not the for (...).

Definition at line 532 of file usb.c.

References DELAY_LONG, DELAY_SHORT, hwload_mode, MSG_SIZE, os_usbsend(), and reset_stop.

```
                                                                                              {
    int total_sent = 0;
    for(int i = 0; i < count; i++){
        // Send each message via the OS function
        while(1){
            DELAY_SHORT(kb);
            int res = os_usbsend(kb, messages + i * MSG_SIZE, 0, file, line);
            if(res == 0)
                return 0;
            else if(res != -1){
                total_sent += res;
                break;
            }
            // Stop immediately if the program is shutting down or hardware load is set to tryonce
            if(reset_stop || hwload_mode != 2)
                return 0;
            // Retry as long as the result is temporary failure
            DELAY_LONG(kb);
        }
    }
    return total_sent;
}
```

Here is the call graph for this function:



5.37.1.5 int closeusb (usbdevice * *kb*)

closeusb Close a USB device and remove device entry.

An imutex lock ensures first of all, that no communication is currently running from the viewpoint of the driver to the user input device (ie the virtual driver with which characters or mouse movements are sent from the daemon to the operating system as inputs).

If the **kb** has an acceptable value = 0, the index of the device is looked for and with this index os_inputclose() is called. After this no more characters can be sent to the operating system.

Then the connection to the usb device is capped by os_closeusb().

Todo What is not yet comprehensible is the call to updateconnected() BEFORE os_closeusb(). Should that be in the other sequence? Or is updateconnected() not displaying the connected usb devices, but the representation which uinput devices are loaded? Questions about questions ...

If there is no valid **handle**, only updateconnected() is called. We are probably trying to disconnect a connection under construction. Not clear.

The cmd pipe as well as all open notify pipes are deleted via rmdevpath ().

This means that nothing can happen to the input path - so the device-specific imutex is unlocked again and remains unlocked.

Also the dmutex is unlocked now, but only to join the thread, which was originally taken under **kb->thread** (which started with _setupusb()) with pthread_join() again. Because of the closed devices that thread would have to quit sometime

See Also

the hack note with rmdevpath())

As soon as the thread is caught, the dmutex is locked again, which is what I do not understand yet: What other thread can do usb communication now?

If the vtabel exists for the given kb (why not? It seems to have race conditions here!!), via the vtable the actually device-specific, but still everywhere identical freeprofile() is called. This frees areas that are no longer needed. Then the **usbdevice** structure in its array is set to zero completely.

Error handling is rather unusual in closeusb(); Everything works (no matter what the called functions return), and closeusb() always returns zero (success).

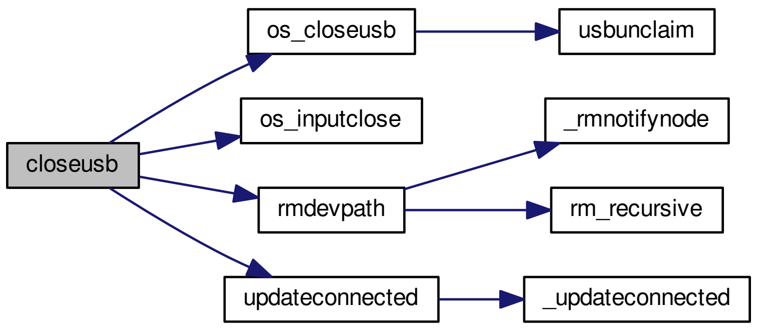
Definition at line 673 of file usb.c.

References ckb_info, devpath, dmutex, usbdevice::handle, imutex, INDEX_OF, keyboard, os_closeusb(), os_-inputclose(), rmdevpath(), usbdevice::thread, updateconnected(), and usbdevice::vtable.

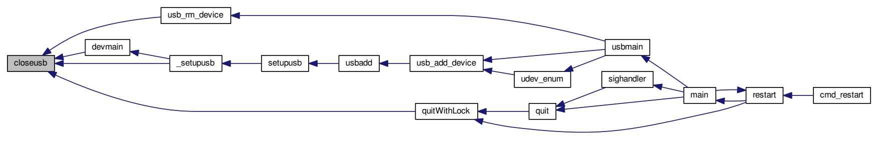
Referenced by _setupusb(), devmain(), quitWithLock(), and usb_rm_device().

```
673                             {
674     pthread_mutex_lock(imutex(kb));
675     if(kb->handle){
676         int index = INDEX_OF(kb, keyboard);
677         ckb_info("Disconnecting %s%d\n", devpath, index);
678         os_inputclose(kb);
679         updateconnected();
680         // Close USB device
681         os_closeusb(kb);
682     } else
683         updateconnected();
684     rmdevpath(kb);
685 
686     // Wait for thread to close
687     pthread_mutex_unlock(imutex(kb));
688     pthread_mutex_unlock(dmutex(kb));
689     pthread_join(kb->thread, 0);
690     pthread_mutex_lock(dmutex(kb));
691 
692     // Delete the profile and the control path
693     if(!kb->vtable)
694         return 0;
695     kb->vtable->freeprofile(kb);
696     memset(kb, 0, sizeof(usbdevice));
697     return 0;
698 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.37.1.6 static void* devmain (**usbdevice** * *kb*) [static]

devmain is called by _setupusb

Parameters

kb	the pointer to the device. Even if it has the name kb, it is valid also for a mouse (the whole driver seems to be implemented first for a keyboard).

Returns

always a nullptr

Synchronization

The syncing via mutexes is interesting:

1. *imutex* (the Input mutex)

   This one is locked in `setupusb()`. That function does only two things: Locking the mutex and trying to start a thread at `_setupusb()`. _setupusb() unlocks *imutex* after getting some buffers and initalizing internal structures from the indicators (this function often gets problems with error messages like "unable to read indicators" or "Timeout bla blubb").

Warning

have a look at *updateindicators()* later.
if creating the thread is not successful, the *imutex* remains blocked. Have a look at setupusb() later.

2. *dmutex* (the Device mutex)

   This one is very interesting, because it is handled in devmain(). It seems that it is locked only in *_ledthread()*, which is a thread created in *os_setupindicators()*. os_setupindicators() again is called in *_setupusb()* long before calling devmain(). So this mutex is locked when we start the function as the old comment says.

   Before reading from the FIFO and direct afterwards an unlock..lock sequence is implemented here. Even if only the function readlines() should be surrounded by the unlock..lock, the variable definition of the line pointer is also included here. Not nice, but does not bother either. Probably the Unlock..lock is needed so that now another process can change the control structure *linectx* while we wait in readlines().

   Todo Hope to find the need for dmutex usage later.
   Should this function be declared as pthread_t∗ function, because of the defintion of pthread-create? But void∗ works also...

Attention

dmutex should still be locked when this is called

First a *readlines_ctx* buffer structure is initialized by `readlines_ctx_init()`.

After some setup functions, beginning in _setupusb() which has called devmain(), we read the command input--Fifo designated to that device in an endless loop. This loop has two possible exits (plus reaction to signals, not mentioned here).

If the reading via readlines() is successful (we might have read multiple lines), the interpretation is done by readcmd() iff the connection to the device is still available (checked via IS_CONNECTED(kb)). This is true if the kb-structure has a handle and an event pointer both != Null). If not, the loop is left (the first exit point).

if nothing is in the line buffer (some magic interrupt?), continue in the endless while without any reaction.

Todo readcmd() gets a **line**, not **lines**. Have a look on that later.
Is the condition IS_CONNECTED valid? What functions change the condititon for the macro?

If interpretation and communication with the usb device got errors, they are signalled by readcmd() (non zero retcode). In this case the usb device is closed via closeusb() and the endless loop is left (the second exit point).

After leaving the endless loop the readlines-ctx structure and its buffers are freed by readlines_ctx_free().

Definition at line 135 of file usb.c.

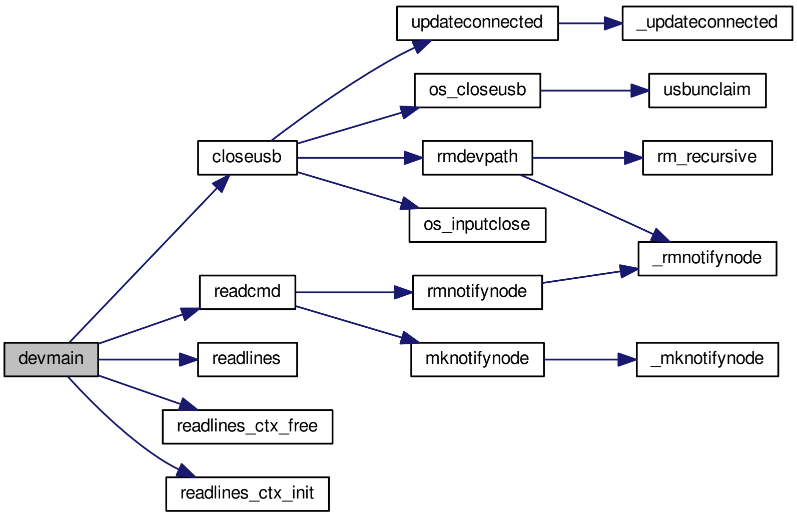
References closeusb(), dmutex, usbdevice::infifo, IS_CONNECTED, readcmd(), readlines(), readlines_ctx_free(), and readlines_ctx_init().

Referenced by _setupusb().

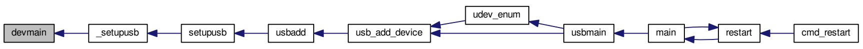
```
135                                 {
137     int kbfifo = kb->infifo - 1;
140     readlines_ctx linectx;
141     readlines_ctx_init(&linectx);
146     while(1){
153         pthread_mutex_unlock(dmutex(kb));
154         // Read from FIFO
155         const char* line;
156         int lines = readlines(kbfifo, linectx, &line);
157         pthread_mutex_lock(dmutex(kb));
158         // End thread when the handle is removed
159         if(!IS_CONNECTED(kb))
160             break;
164         if(lines){
167             if(readcmd(kb, line)){
173                 // USB transfer failed; destroy device
174                 closeusb(kb);
175                 break;
```

```
            }
        }
    }
    pthread_mutex_unlock(dmutex(kb));
    readlines_ctx_free(linectx);
    return 0;
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.37.1.7 static const **devcmd**∗ get_vtable (short *vendor,* short *product*) [static]

get_vtable returns the correct vtable pointer

Parameters

vendor	short usb vendor ID
product	short usb product ID

Returns

Depending on the type and model, the corresponding vtable pointer is returned (see below)

At present, we have three different vtables:

- `vtable_mouse` is used for all mouse types. This may be wrong with some newer mice?
- `vtable_keyboard` is used for all RGB Keyboards.
- `vtable_keyboard_nonrgb` for all the rest.

Todo Is the last point really a good decision and always correct?

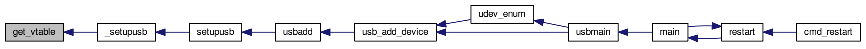
Definition at line 102 of file usb.c.

References IS_MOUSE, IS_RGB, vtable_keyboard, vtable_keyboard_nonrgb, and vtable_mouse.

Referenced by _setupusb().

```
102                                                              {
103     return IS_MOUSE(vendor, product) ? &vtable_mouse :
      IS_RGB(vendor, product) ? &vtable_keyboard : &
      vtable_keyboard_nonrgb;
104 }
```

Here is the caller graph for this function:



5.37.1.8 const char∗ product_str (short *product*)

product_str returns a condensed view on what type of device we have.

At present, various models and their properties are known from corsair products. Some models differ in principle (mice and keyboards), others differ in the way they function (for example, RGB and non RGB), but they are very similar.

Here, only the first point is taken into consideration and we return a unified model string. If the model is not known with its number, *product_str* returns an empty string.

The model numbers and corresponding strings wwith the numbers in hex-string are defined in `usb.h`

At present, this function is used to initialize `kb->name` and to give information in debug strings.

Attention

The combinations below have to fit to the combinations in the macros mentioned above. So if you add a device with a new number, change both.

Todo There are macros defined in usb.h to detect all the combinations below. the only difference is the parameter: The macros need the *kb*∗, product_str() needs the *product ID*

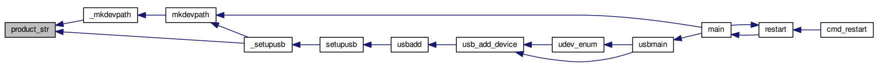
Definition at line 70 of file usb.c.

References P_HARPOON, 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_P-RO, P_STRAFE, and P_STRAFE_NRGB.

Referenced by _mkdevpath(), and _setupusb().

```
70                                       {
71     if(product == P_K95 || product == P_K95_NRGB || product ==
      P_K95_PLATINUM)
72         return "k95";
73     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)
74         return "k70";
75     if(product == P_K65 || product == P_K65_NRGB || product ==
      P_K65_LUX || product == P_K65_RFIRE)
76         return "k65";
77     if(product == P_STRAFE || product == P_STRAFE_NRGB)
78          return "strafe";
79      if(product == P_M65 || product == P_M65_PRO)
80          return "m65";
81      if(product == P_SABRE_O || product == P_SABRE_L || product ==
     P_SABRE_N || product == P_SABRE_O2 || product == P_HARPOON)
82          return "sabre";
83      if(product == P_SCIMITAR || product == P_SCIMITAR_PRO)
84          return "scimitar";
85      return "";
86 }
```

Here is the caller graph for this function:



5.37.1.9 int revertusb (usbdevice ∗ *kb*)

revertusb sets a given device to inactive (hardware controlled) mode if not a fw-ugrade is indicated

First is checked, whether a firmware-upgrade is indicated for the device. If so, revertusb() returns 0.

Todo Why is this useful? Are there problems seen with deactivating a device with older fw-version??? Why isn't this an error indicating reason and we return success (0)?

Anyway, the following steps are similar to some other procs, dealing with low level usb handling:

- If we do not have an RGB device, a simple setting to Hardware-mode (NK95_HWON) is sent to the device via n95cmd().

  Todo The return value of nk95cmd() is ignored (but sending the ioctl may produce an error and _nk95_cmd will indicate this), instead revertusb() returns success in any case.

- If we have an RGB device, setactive() is called with second param active = false. That function will have a look on differences between keyboards and mice.

  More precisely setactive() is just a macro to call via the kb->vtable enties either the active() or the idle() function where the vtable points to. setactive() may return error indications. If so, revertusb() returns -1, otherwise 0 in any other case.

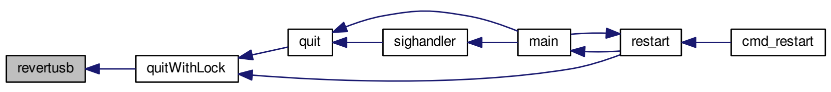
Definition at line 407 of file usb.c.

References FEAT_RGB, HAS_FEATURES, NEEDS_FW_UPDATE, NK95_HWON, nk95cmd, and setactive.

Referenced by quitWithLock().

```
407                             {
408     if(NEEDS_FW_UPDATE(kb))
409         return 0;
410     if(!HAS_FEATURES(kb, FEAT_RGB)){
411         nk95cmd(kb, NK95_HWON);
412         return 0;
413     }
414     if(setactive(kb, 0))
415         return -1;
416     return 0;
417 }
```

Here is the caller graph for this function:



5.37.1.10 void setupusb (usbdevice ∗ *kb*)

setupusb starts a thread with kb as parameter and _setupusb() as entrypoint.

Set up a USB device after its handle is open. Spawns a new thread _setupusb() with standard parameter kb. dmutex must be locked prior to calling this function. The function will unlock it when finished. In kb->thread the thread id is mentioned, because closeusb() needs this info for joining that thread again.

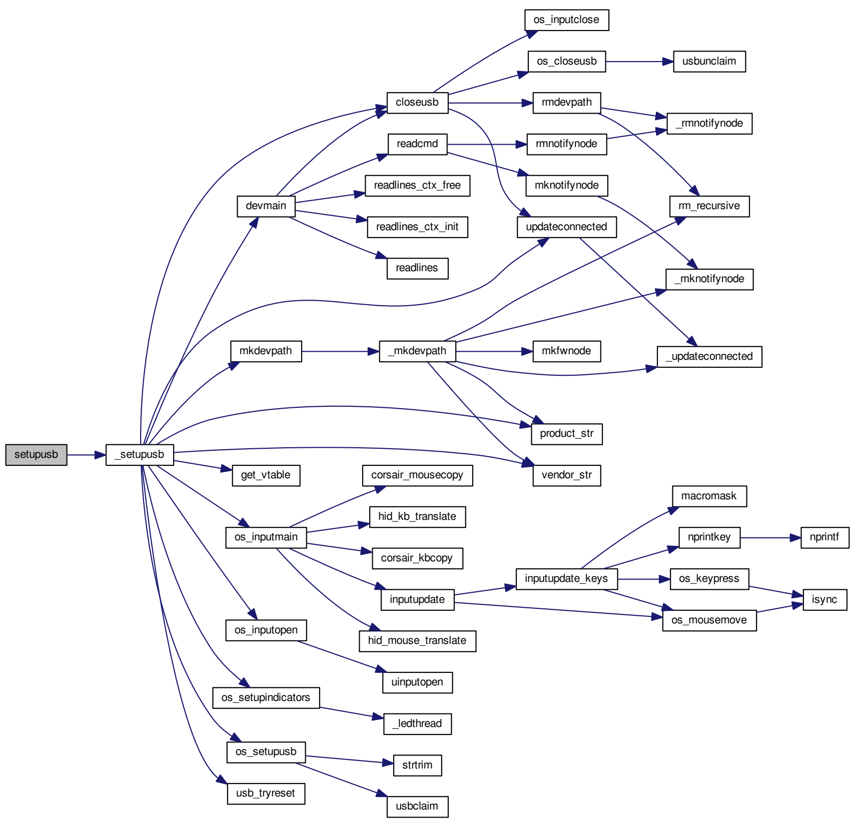
Definition at line 386 of file usb.c.

References _setupusb(), ckb_err, imutex, and usbdevice::thread.

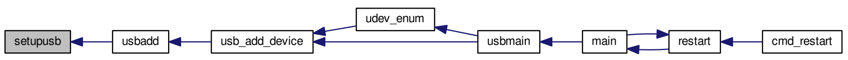
Referenced by usbadd().

```
386                             {
387     pthread_mutex_lock(imutex(kb));
388     if(pthread_create(&kb->thread, 0, _setupusb, kb))
389         ckb_err("Failed to create USB thread\n");
390 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.37.1.11 int usb_tryreset (**usbdevice** ∗ *kb*)

usb_tryreset does what the name means: Try to reset the usb via resetusb()

This function is called if an usb command ran into an error in case of one of the following two situations:

- When setting up a new usb device and the start() function got an error (

  See Also

  _setupusb())

- If upgrading to a new firmware gets an error (

  See Also

  cmd_fwupdate()).

  The previous action which got the error will NOT be re-attempted.

In an endless loop usb_tryreset() tries to reset the given usb device via the macro resetusb().

This macro calls _resetusb() with debugging information.

_resetusb() sends a command via the operating system dependent function os_resetusb() and - if successful - reinitializes the device. os_resetusb() returns -2 to indicate a broken device and all structures should be removed for it.

In that case, the loop is terminated, an error message is produced and usb_tryreset() returns -1.

In case resetusb() has success, the endless loop is left via a return 0 (success).

If the return value from resetusb() is -1, the loop is continued with the next try.

If the global variable **reset_stop** is set directly when the function is called or after each try, usb_tryreset() stops working and returns -1.

Todo Why does usb_tryreset() hide the information returned from resetusb()? Isn't it needed by the callers?

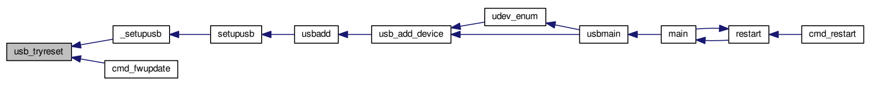
Definition at line 465 of file usb.c.

References ckb_err, ckb_info, reset_stop, and resetusb.

Referenced by _setupusb(), and cmd_fwupdate().

```
                                  {
    if(reset_stop)
        return -1;
    ckb_info("Attempting reset...\n");
    while(1){
        int res = resetusb(kb);
        if(!res){
            ckb_info("Reset success\n");
            return 0;
        }
        if(res == -2 || reset_stop)
            break;
    }
    ckb_err("Reset failed. Disconnecting.\n");
    return -1;
}
```

Here is the caller graph for this function:



5.37.1.12 const char∗ vendor_str (short *vendor*)

uncomment the following Define to see USB packets sent to the device

vendor_str returns "corsair" iff the given *vendor* argument is equal to *V_CORSAIR* (0x1bc) else it returns ""

Attention

There is also a string defined V_CORSAIR_STR, which returns the device number as string in hex "1b1c".

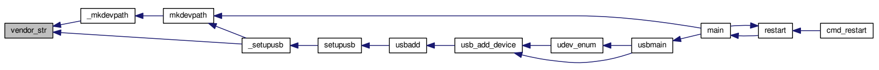
Definition at line 43 of file usb.c.

References V_CORSAIR.

Referenced by _mkdevpath(), and _setupusb().

```
43                                  {
44     if(vendor == V_CORSAIR)
45         return "corsair";
46     return "";
47 }
```

Here is the caller graph for this function:



5.37.2 Variable Documentation

5.37.2.1 int features_mask = -1

features_mask Mask of features to exclude from all devices

That bit mask ist set to enable all (-1). When interpreting the input parameters, some of these bits can be cleared.

At the moment binding, notifying and mouse-acceleration can be disabled via command line.

Have a look at *main()* in main.c for details.

Definition at line 35 of file usb.c.

Referenced by _setupusb(), and main().

5.37.2.2 int hwload_mode

Definition at line 7 of file device.c.

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

5.37.2.3 volatile int reset_stop = 0

reset_stop is boolean: Reset stopper for when the program shuts down.

Is set only by *quit()* to true (1) to inform several usb_∗ functions to end their loops and tries.

Definition at line 25 of file usb.c.

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

5.37.2.4 pthread_mutex_t usbmutex = PTHREAD_MUTEX_INITIALIZER

usbmutex is a never referenced mutex!

Todo We should have a look why this mutex is never used.

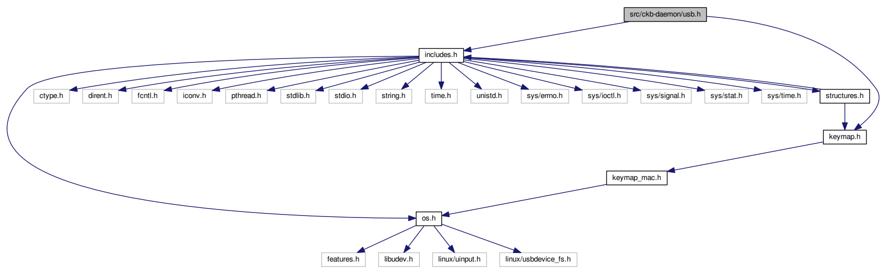
Definition at line 17 of file usb.c.

5.38 src/ckb-daemon/usb.h File Reference

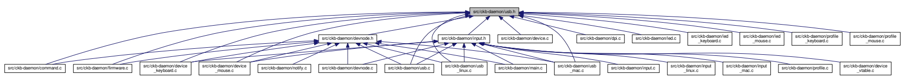
Definitions for using USB interface.

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

Include dependency graph for usb.h:



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


src/ckb-daemon/usb.h
src/ckb-daemon/devnode.h
src/ckb-daemon/input.h
src/ckb-daemon/device.c
src/ckb-daemon/dpi.c
src/ckb-daemon/led.c
src/ckb-daemon/led_keyboard.c
src/ckb-daemon/led_mouse.c
src/ckb-daemon/profile_keyboard.c
src/ckb-daemon/profile_mouse.c
src/ckb-daemon/command.c
src/ckb-daemon/firmware.c
src/ckb-daemon/device_keyboard.c
src/ckb-daemon/device_mouse.c
src/ckb-daemon/notify.c
src/ckb-daemon/devnode.c
src/ckb-daemon/usb.c
src/ckb-daemon/usb_linux.c
src/ckb-daemon/main.c
src/ckb-daemon/usb_mac.c
src/ckb-daemon/input.c
src/ckb-daemon/input_linux.c
src/ckb-daemon/input_mac.c
src/ckb-daemon/profile.c
src/ckb-daemon/device_vtable.c

Macros

- #define V_CORSAIR 0x1b1c

  For the following Defines please see "Detailed Description".
- #define P_K65 0x1b17
- #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 P_HARPOON 0x1b3c /∗ Harpoon test ∗/
- #define P_HARPOON_STR "1b3c"
- #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 || (kb)->product == P_HARPOON))
- #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))

  RGB vs non-RGB test (note: non-RGB Strafe is still considered "RGB" in that it shares the same protocol. The difference is denoted with the "monochrome" feature).
- #define IS_MONOCHROME(vendor, product) ((vendor) == (V_CORSAIR) && (product) == (P_STRAFE_N-RGB))

  The difference between non RGB and monochrome is, that monochrome has lights, but just in one color. nonRGB has no lights. Change this if new ***monochrome*** *devices are added.*
- #define IS_RGB_DEV(kb) IS_RGB((kb)->vendor, (kb)->product)

  For calling with a usbdevice∗, vendor and product are extracted and IS_RGB() is returned.
- #define IS_MONOCHROME_DEV(kb) IS_MONOCHROME((kb)->vendor, (kb)->product)

  For calling with a usbdevice∗, vendor and product are extracted and IS_MONOCHROME() is returned.
- #define IS_FULLRANGE(kb) (IS_RGB((kb)->vendor, (kb)->product) && (kb)->product != P_K65 && (kb)->product != P_K70 && (kb)->product != P_K95)

  Full color range (16.8M) vs partial color range (512)
- #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)))

  Mouse vs keyboard test.
- #define IS_MOUSE_DEV(kb) IS_MOUSE((kb)->vendor, (kb)->product)

  For calling with a usbdevice∗, vendor and product are extracted and IS_MOUSE() is returned.
- #define DELAY_SHORT(kb) usleep((int)(kb)->usbdelay ∗ 1000)

  USB delays for when the keyboards get picky about timing That was the original comment, but it is used anytime. The short delay is used before any send or receive.
- #define DELAY_MEDIUM(kb) usleep((int)(kb)->usbdelay ∗ 10000)

  the medium delay is used after sending a command before waiting for the answer.
- #define DELAY_LONG(kb) usleep(100000)

  The longest delay takes place where something went wrong (eg when resetting the device)
- #define USB_DELAY_DEFAULT 5

  This constant is used to initialize ***kb->usbdelay****. It is used in many places (see macros above) but often also overwritten to the fixed value of 10. Pure Hacker code.*
- #define resetusb(kb) _resetusb(kb, __FILE_NOPATH__, __LINE__)

  resetusb() is just a macro to call _resetusb() with debuggin constants (file, lineno)
- #define usbsend(kb, messages, count) _usbsend(kb, messages, count, __FILE_NOPATH__, __LINE__)

  usbsend macro is used to wrap _usbsend() with debugging information (file and lineno)
- #define usbrecv(kb, out_msg, in_msg) _usbrecv(kb, out_msg, in_msg, __FILE_NOPATH__, __LINE__)

  usbrecv macro is used to wrap _usbrecv() with debugging information (file and lineno)
- #define nk95cmd(kb, command) _nk95cmd(kb, (command) >> 16 & 0xFF, (command) & 0xFFFF, __FILE_-NOPATH__, __LINE__)

  nk95cmd() macro is used to wrap _nk95cmd() with debugging information (file and lineno). the command structure is different:
  Just the bits 23..16 are used as bits 7..0 for bRequest
  Bits 15..0 are used as wValue
- #define NK95_HWOFF 0x020030

  Hardware-specific commands for the K95 nonRGB,.
- #define NK95_HWON 0x020001

  Hardware playback on.
- #define NK95_M1 0x140001

  Switch to mode 1.

- #define NK95_M2 0x140002

  Switch to mode 2.
- #define NK95_M3 0x140003

  Switch to mode 3.

Functions

- const char ∗ vendor_str (short vendor)

  uncomment the following Define to see USB packets sent to the device
- const char ∗ product_str (short product)

  product_str returns a condensed view on what type of device we have.
- int usbmain ()

  Start the USB main loop. Returns program exit code when finished.
- void usbkill ()

  Stop the USB system.
- void setupusb (usbdevice ∗kb)

  setupusb starts a thread with kb as parameter and _setupusb() as entrypoint.
- int os_setupusb (usbdevice ∗kb)

  os_setupusb OS-specific setup for a specific usb device.
- void ∗ os_inputmain (void ∗context)

  os_inputmain is run in a separate thread and will be detached from the main thread, so it needs to clean up its own resources.
- int revertusb (usbdevice ∗kb)

  revertusb sets a given device to inactive (hardware controlled) mode if not a fw-ugrade is indicated
- int closeusb (usbdevice ∗kb)

  closeusb Close a USB device and remove device entry.
- void os_closeusb (usbdevice ∗kb)

  os_closeusb unclaim it, destroy the udev device and clear data structures at kb
- int _resetusb (usbdevice ∗kb, const char ∗file, int line)

  _resetusb Reset a USB device.
- int os_resetusb (usbdevice ∗kb, const char ∗file, int line)

  os_resetusb is the os specific implementation for resetting usb
- int _usbsend (usbdevice ∗kb, const uchar ∗messages, int count, const char ∗file, int line)

  _usbsend send a logical message completely to the given device
- int _usbrecv (usbdevice ∗kb, const uchar ∗out_msg, uchar ∗in_msg, const char ∗file, int line)

  _usbrecv Request data from a USB device by first sending an output packet and then reading the response.
- int os_usbsend (usbdevice ∗kb, const uchar ∗out_msg, int is_recv, const char ∗file, int line)

  os_usbsend sends a data packet (MSG_SIZE = 64) Bytes long
- int os_usbrecv (usbdevice ∗kb, uchar ∗in_msg, const char ∗file, int line)

  os_usbrecv receives a max MSGSIZE long buffer from usb device
- void os_sendindicators (usbdevice ∗kb)

  os_sendindicators update the indicators for the special keys (Numlock, Capslock and what else?)
- int _nk95cmd (usbdevice ∗kb, uchar bRequest, ushort wValue, const char ∗file, int line)

  _nk95cmd If we control a non RGB keyboard, set the keyboard via ioctl with usbdevfs_ctrltransfer
- int usb_tryreset (usbdevice ∗kb)

  usb_tryreset does what the name means: Try to reset the usb via resetusb()

5.38.1 Detailed Description

Vendor/product codes

The list of defines in the first part of the file describes the various types of equipment from Corsair and summarizes them according to specific characteristics.

Each device type is described with two defines:

- On the one hand the device ID with which the device can be recognized on the USB as a short
- and on the other hand the same representation as a string, but without leading "0x".

First entry-pair is the Provider ID (vendorID) from Corsair.

Block No.	contains	Devices are bundled via
1	The first block contains the K65-like keyboards, regardless of their properties (RGB, ...).	In summary, they can be queried using the macro IS_K65().
2	the K70-like Keyboards with all their configuration types	summarized by IS_K70().
3	the K95 series keyboards	collected with the macro IS_K95().
4	strafe keyboards	IS_STRAFE()
5	M65 mice with and without RGB	IS_M65()
6	The SABRE and HARPOON mice. Maybe this will be divided int two different blocks later because of different nummber of special keys	IS_SABRE()
7	The Scimitar mouse devices	IS_SCIMITAR()

Definition in file usb.h.

5.38.2 Macro Definition Documentation

5.38.2.1 #define DELAY_LONG(*kb*) usleep(100000)

Definition at line 156 of file usb.h.

Referenced by _resetusb(), _setupusb(), _usbrecv(), _usbsend(), cmd_hwload_kb(), cmd_hwload_mouse(), cmd_-hwsave_kb(), and cmd_hwsave_mouse().

5.38.2.2 #define DELAY_MEDIUM(*kb*) usleep((int)(kb)->usbdelay ∗ 10000)

Definition at line 153 of file usb.h.

Referenced by _usbrecv(), and setactive_kb().

5.38.2.3 #define DELAY_SHORT(*kb*) usleep((int)(kb)->usbdelay ∗ 1000)

Definition at line 150 of file usb.h.

Referenced by _usbrecv(), _usbsend(), and updateindicators_kb().

5.38.2.4 #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 139 of file usb.h.

Referenced by readcmd(), and updatergb_kb().

5.38.2.5 #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 49 of file usb.h.

Referenced by has_key().

5.38.2.6 #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 63 of file usb.h.

5.38.2.7 #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 71 of file usb.h.

Referenced by cmd_hwload_kb(), cmd_hwsave_kb(), and has_key().

5.38.2.8 #define IS_M65(*kb*) ((kb)->vendor == V_CORSAIR && ((kb)->product == P_M65 || (kb)->product == P_M65_PRO))

Definition at line 83 of file usb.h.

Referenced by isblack().

5.38.2.9 #define IS_MONOCHROME(*vendor, product*) ((vendor) == (V_CORSAIR) && (product) == (P_STRAFE_NRGB))

Definition at line 130 of file usb.h.

Referenced by _setupusb().

5.38.2.10 #define IS_MONOCHROME_DEV(*kb*) IS_MONOCHROME((kb)->vendor, (kb)->product)

Definition at line 136 of file usb.h.

5.38.2.11 #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 142 of file usb.h.

Referenced by _setupusb(), get_vtable(), has_key(), and os_inputmain().

5.38.2.12 #define IS_MOUSE_DEV(*kb*) IS_MOUSE((kb)->vendor, (kb)->product)

Definition at line 145 of file usb.h.

Referenced by readcmd().

5.38.2.13 #define IS_RGB(*vendor, product*) ((vendor) == (V_CORSAIR) && (product) != (P_K65_NRGB) && (product) != (P_K70_NRGB) && (product) != (P_K95_NRGB))

Definition at line 125 of file usb.h.

Referenced by _setupusb(), get_vtable(), and os_inputmain().

5.38.2.14 #define IS_RGB_DEV(*kb*) IS_RGB((kb)->vendor, (kb)->product)

Definition at line 133 of file usb.h.

5.38.2.15 #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 || (kb)->product == P_HARPOON))

Definition at line 95 of file usb.h.

Referenced by has_key(), loadrgb_mouse(), and savergb_mouse().

5.38.2.16 #define IS_SCIMITAR(*kb*) ((kb)->vendor == V_CORSAIR && ((kb)->product == P_SCIMITAR || (kb)->product == P_SCIMITAR_PRO))

Definition at line 101 of file usb.h.

Referenced by has_key(), loadrgb_mouse(), and savergb_mouse().

5.38.2.17 #define IS_STRAFE(*kb*) ((kb)->vendor == V_CORSAIR && ((kb)->product == P_STRAFE || (kb)->product == P_STRAFE_NRGB))

Definition at line 77 of file usb.h.

Referenced by savergb_kb().

5.38.2.18 #define NK95_HWOFF 0x020030

See Also

`usb2.0 documentation for details`. Set Hardware playback off

Definition at line 302 of file usb.h.

Referenced by start_kb_nrgb().

5.38.2.19 #define NK95_HWON 0x020001

Definition at line 305 of file usb.h.

Referenced by revertusb().

5.38.2.20 #define NK95_M1 0x140001

Definition at line 308 of file usb.h.

Referenced by setmodeindex_nrgb().

5.38.2.21 #define NK95_M2 0x140002

Definition at line 311 of file usb.h.

Referenced by setmodeindex_nrgb().

5.38.2.22 #define NK95_M3 0x140003

Definition at line 314 of file usb.h.

Referenced by setmodeindex_nrgb().

5.38.2.23 #define nk95cmd(*kb, command*) _nk95cmd(kb, (command) $>>$ 16 & 0xFF, (command) & 0xFFFF, __FILE_NOPATH__, __LINE__)

Definition at line 297 of file usb.h.

Referenced by revertusb(), setmodeindex_nrgb(), and start_kb_nrgb().

5.38.2.24 #define P_HARPOON 0x1b3c /* Harpoon test */

Definition at line 93 of file usb.h.

Referenced by product_str().

5.38.2.25 #define P_HARPOON_STR "1b3c"

Definition at line 94 of file usb.h.

5.38.2.26 #define P_K65 0x1b17

Definition at line 41 of file usb.h.

Referenced by product_str().

5.38.2.27 #define P_K65_LUX 0x1b37

Definition at line 45 of file usb.h.

Referenced by product_str().

5.38.2.28 #define P_K65_LUX_STR "1b37"

Definition at line 46 of file usb.h.

5.38.2.29 #define P_K65_NRGB 0x1b07

Definition at line 43 of file usb.h.

Referenced by product_str().

5.38.2.30 #define P_K65_NRGB_STR "1b07"

Definition at line 44 of file usb.h.

5.38.2.31 #define P_K65_RFIRE 0x1b39

Definition at line 47 of file usb.h.

Referenced by product_str().

5.38.2.32 #define P_K65_RFIRE_STR "1b39"

Definition at line 48 of file usb.h.

5.38.2.33 #define P_K65_STR "1b17"

Definition at line 42 of file usb.h.

5.38.2.34 #define P_K70 0x1b13

Definition at line 51 of file usb.h.

Referenced by product_str().

5.38.2.35 #define P_K70_LUX 0x1b33

Definition at line 55 of file usb.h.

Referenced by product_str().

5.38.2.36 #define P_K70_LUX_NRGB 0x1b36

Definition at line 57 of file usb.h.

Referenced by product_str().

5.38.2.37 #define P_K70_LUX_NRGB_STR "1b36"

Definition at line 58 of file usb.h.

5.38.2.38 #define P_K70_LUX_STR "1b33"

Definition at line 56 of file usb.h.

5.38.2.39 #define P_K70_NRGB 0x1b09

Definition at line 53 of file usb.h.

Referenced by product_str().

5.38.2.40 #define P_K70_NRGB_STR "1b09"

Definition at line 54 of file usb.h.

5.38.2.41 #define P_K70_RFIRE 0x1b38

Definition at line 59 of file usb.h.

Referenced by product_str().

5.38.2.42 #define P_K70_RFIRE_NRGB 0x1b3a

Definition at line 61 of file usb.h.

Referenced by product_str().

5.38.2.43 #define P_K70_RFIRE_NRGB_STR "1b3a"

Definition at line 62 of file usb.h.

5.38.2.44 #define P_K70_RFIRE_STR "1b38"

Definition at line 60 of file usb.h.

5.38.2.45 #define P_K70_STR "1b13"

Definition at line 52 of file usb.h.

5.38.2.46 #define P_K95 0x1b11

Definition at line 65 of file usb.h.

Referenced by product_str().

5.38.2.47 #define P_K95_NRGB 0x1b08

Definition at line 67 of file usb.h.

Referenced by _nk95cmd(), and product_str().

5.38.2.48 #define P_K95_NRGB_STR "1b08"

Definition at line 68 of file usb.h.

5.38.2.49 #define P_K95_PLATINUM 0x1b2d

Definition at line 69 of file usb.h.

Referenced by product_str().

5.38.2.50 #define P_K95_PLATINUM_STR "1b2d"

Definition at line 70 of file usb.h.

5.38.2.51 #define P_K95_STR "1b11"

Definition at line 66 of file usb.h.

5.38.2.52 #define P_M65 0x1b12

Definition at line 79 of file usb.h.

Referenced by product_str().

5.38.2.53 #define P_M65_PRO 0x1b2e

Definition at line 81 of file usb.h.

Referenced by product_str().

5.38.2.54 #define P_M65_PRO_STR "1b2e"

Definition at line 82 of file usb.h.

5.38.2.55 #define P_M65_STR "1b12"

Definition at line 80 of file usb.h.

5.38.2.56 #define P_SABRE_L 0x1b19 /∗ laser ∗/

Definition at line 87 of file usb.h.

Referenced by product_str().

5.38.2.57 #define P_SABRE_L_STR "1b19"

Definition at line 88 of file usb.h.

5.38.2.58 #define P_SABRE_N 0x1b2f /∗ new? ∗/

Definition at line 89 of file usb.h.

Referenced by product_str().

5.38.2.59 #define P_SABRE_N_STR "1b2f"

Definition at line 90 of file usb.h.

5.38.2.60 #define P_SABRE_O 0x1b14 /∗ optical ∗/

Definition at line 85 of file usb.h.

Referenced by product_str().

5.38.2.61 #define P_SABRE_O2 0x1b32 /∗ Observed on a CH-9000111-EU model SABRE ∗/

Definition at line 91 of file usb.h.

Referenced by product_str().

5.38.2.62 #define P_SABRE_O2_STR "1b32"

Definition at line 92 of file usb.h.

5.38.2.63 #define P_SABRE_O_STR "1b14"

Definition at line 86 of file usb.h.

5.38.2.64 #define P_SCIMITAR 0x1b1e

Definition at line 97 of file usb.h.

Referenced by product_str().

5.38.2.65 #define P_SCIMITAR_PRO 0x1b3e

Definition at line 99 of file usb.h.

Referenced by product_str().

5.38.2.66 #define P_SCIMITAR_PRO_STR "1b3e"

Definition at line 100 of file usb.h.

5.38.2.67 #define P_SCIMITAR_STR "1b1e"

Definition at line 98 of file usb.h.

5.38.2.68 #define P_STRAFE 0x1b20

Definition at line 73 of file usb.h.

Referenced by product_str().

5.38.2.69 #define P_STRAFE_NRGB 0x1b15

Definition at line 75 of file usb.h.

Referenced by product_str().

5.38.2.70 #define P_STRAFE_NRGB_STR "1b15"

Definition at line 76 of file usb.h.

5.38.2.71 #define P_STRAFE_STR "1b20"

Definition at line 74 of file usb.h.

5.38.2.72 #define resetusb(*kb*) _resetusb(kb, __FILE_NOPATH__, __LINE__)

Definition at line 215 of file usb.h.

Referenced by usb_tryreset().

5.38.2.73 #define USB_DELAY_DEFAULT 5

Definition at line 161 of file usb.h.

Referenced by _setupusb(), and start_dev().

5.38.2.74 #define usbrecv(*kb, out_msg, in_msg*) _usbrecv(kb, out_msg, in_msg, __FILE_NOPATH__, __LINE__)

Parameters

kb	THE usbdevice∗
IN]	out_msg What information does the caller want from the device?
OUT]	in_msg Here comes the answer; The names represent the usb view, not the view of this function! So INput from usb is OUTput of this function.

Definition at line 257 of file usb.h.

Referenced by cmd_hwload_kb(), cmd_hwload_mouse(), getfwversion(), hwloadmode(), loaddpi(), loadrgb_kb(), and loadrgb_mouse().

5.38.2.75 #define usbsend(*kb,* *messages,* *count*) _usbsend(kb, messages, count, __FILE_NOPATH__, __LINE__)

Parameters

kb	THE usbdevice∗
IN]	messages a Pointer to the first byte of the logical message
IN]	count how many MSG_SIZE buffers is the logical message long?

Definition at line 240 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().

5.38.2.76 #define V_CORSAIR 0x1b1c

Warning

> When adding new devices please update src/ckb/fwupgradedialog.cpp as well.
> It should contain the same vendor/product IDs for any devices supporting firmware updates.
> In the same way, all other corresponding files have to be supplemented or modified: Currently known for this are **usb_linux.c** and **usb_mac.c**

Definition at line 38 of file usb.h.

Referenced by usb_add_device(), and vendor_str().

5.38.2.77 #define V_CORSAIR_STR "1b1c"

Definition at line 39 of file usb.h.

Referenced by udev_enum(), and usb_add_device().

5.38.3 Function Documentation

5.38.3.1 int _nk95cmd (usbdevice ∗ *kb,* uchar *bRequest,* ushort *wValue,* const char ∗ *file,* int *line*)

Parameters

kb	THE usbdevice∗
bRequest	the byte array with the usb request
wValue	a usb wValue

file	for error message
line	for error message

Returns

1 (true) on failure, 0 (false) on success.

To send control packets to a non RGB non color K95 Keyboard, use this function. Normally it is called via the nk95cmd() macro.

If it is the wrong device for which the function is called, 0 is returned and nothing done. Otherwise a usbdevfs_-ctrltransfer structure is filled and an USBDEVFS_CONTROL ioctl() called.

bRequest-Type	bRequest	wValue	EP	size	Timeout	data
0x40	see table below to switch hardware-modus at Keyboard	wValue	device	MSG_SIZE	5ms	the message buffer pointer
Host to Device, Type=Vendor, Recipi-ent=Device	bRequest parameter	given wValue Parameter	device 0	0 data to write	5000	null

If a 0 or a negative error number is returned by the ioctl, an error message is shown depending on the errno or "No data written" if retval was 0. In either case 1 is returned to indicate the error. If the ioctl returned a value $>$ 0, 0 is returned to indicate no error.

Currently the following combinations for bRequest and wValue are used:

Device	what it might to do	constant	bRequest	wValue
non RGB Keyboard	set HW-modus on (leave the ckb driver)	HWON	0x0002	0x0030
non RGB Keyboard	set HW-modus off (initialize the ckb driver)	HWOFF	0x0002	0x0001
non RGB Keyboard	set light modus M1 in single-color keyboards	NK95_M1	0x0014	0x0001
non RGB Keyboard	set light modus M2 in single-color keyboards	NK95_M2	0x0014	0x0002
non RGB Keyboard	set light modus M3 in single-color keyboards	NK95_M3	0x0014	0x0003

See Also

usb.h

Definition at line 187 of file usb_linux.c.

References ckb_err_fn, usbdevice::handle, P_K95_NRGB, and usbdevice::product.

```
                                                                                                    {
        if(kb->product != P_K95_NRGB)
            return 0;
```

```
    struct usbdevfs_ctrltransfer transfer = { 0x40, bRequest, wValue, 0, 0, 5000, 0 };
    int res = ioctl(kb->handle - 1, USBDEVFS_CONTROL, &transfer);
    if(res <= 0){
        ckb_err_fn("%s\n", file, line, res ? strerror(errno) : "No data written");
        return 1;
    }
    return 0;
}
```

5.38.3.2 int _resetusb (usbdevice ∗ *kb,* const char ∗ *file,* int *line*)

Parameters

kb	THE usbdevice∗
file	filename for error messages
line	line where it is called for error messages

Returns

Returns 0 on success, -1 if device should be removed

_resetusb Reset a USB device.

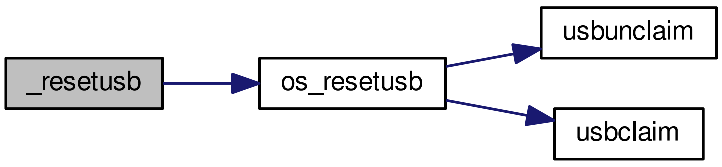
First reset the device via os_resetusb() after a long delay (it may send something to the host). If this worked (retval == 0), give the device another long delay Then perform the initialization via the device specific start() function entry in kb->vtable and if this is successful also, return the result of the device depenten updatergb() with force=true.

Definition at line 426 of file usb.c.

References usbdevice::active, DELAY_LONG, os_resetusb(), and usbdevice::vtable.

```
                                                         {
    // Perform a USB reset
    DELAY_LONG(kb);
    int res = os_resetusb(kb, file, line);
    if(res)
        return res;
    DELAY_LONG(kb);
    // Re-initialize the device
    if(kb->vtable->start(kb, kb->active) != 0)
        return -1;
    if(kb->vtable->updatergb(kb, 1) != 0)
        return -1;
    return 0;
}
```

Here is the call graph for this function:



5.38.3.3 int _usbrecv (usbdevice ∗ *kb,* const uchar ∗ *out_msg,* uchar ∗ *in_msg,* const char ∗ *file,* int *line*)

Parameters

	kb	THE usbdevice∗
	IN]	out_msg What information does the caller want from the device?
	OUT]	in_msg Here comes the answer; The names represent the usb view, not the view of this function! So INput from usb is OUTput of this function.
	IN]	file for debugging
	IN]	line for debugging
`in`	*reset_stop*	global variable is read

Returns

number of bytes read or zero on failure.

_usbrecv Request data from a USB device by first sending an output packet and then reading the response.

To fully understand this, you need to know about usb: All control is at the usb host (the CPU). If the device wants to communicate something to the host, it must wait for the host to ask. The usb protocol defines the cycles and periods in which actions are to be taken.

So in order to receive a data packet from the device, the host must first send a send request.

This is done by _usbrecv() in the first block by sending the MSG_SIZE large data block from **out_msg** via os_usbsend() as it is a machine depending implementation. The usb target device is as always determined over kb.

For os_usbsend() to know that it is a receive request, the **is_recv** parameter is set to true (1). With this, os_usbsend () generates a control package for the hardware, not a data packet.

If sending of the control package is not successful, a maximum of 5 times the transmission is repeated (including the first attempt). If a non-cancelable error is signaled or the drive is stopped via reset_stop, _usbrecv() immediately returns 0.

After this, the function waits for the requested response from the device using os_usbrecv ().

os_usbrecv() returns 0, -1 or something else.

Zero signals a serious error which is not treatable and _usbrecv() also returns 0.

-1 means that it is a treatable error - a timeout for example - and therefore the next transfer attempt is started after a long pause (DELAY_LONG) if not reset_stop or the wrong hwload_mode require a termination with a return value of 0.

After 5 attempts, _usbrecv () returns and returns 0 as well as an error message.

When data is received, the number of received bytes is returned. This should always be MSG_SIZE, but os_usbrecv() can also return less. It should not be more, because then there would be an unhandled buffer overflow, but it could be less. This would be signaled in os_usbrecv () with a message.

The buffers behind **out_msg** and **in_msg** are MSG_SIZE at least (currently 64 Bytes). More is ok but useless, less brings unpredictable behavior.

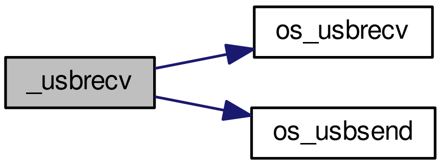
Definition at line 599 of file usb.c.

References ckb_err_fn, DELAY_LONG, DELAY_MEDIUM, DELAY_SHORT, hwload_mode, os_usbrecv(), os_usbsend(), and reset_stop.

```
599                                                                                      {
600     // Try a maximum of 3 times
601     for(int try = 0; try < 5; try++){
602         // Send the output message
603         DELAY_SHORT(kb);
604         int res = os_usbsend(kb, out_msg, 1, file, line);
605         if(res == 0)
606             return 0;
607         else if(res == -1){
608             // Retry on temporary failure
609             if(reset_stop)
610                 return 0;
611             DELAY_LONG(kb);
```

```
            continue;
        }
        // Wait for the response
        DELAY_MEDIUM(kb);
        res = os_usbrecv(kb, in_msg, file, line);
        if(res == 0)
            return 0;
        else if(res != -1)
            return res;
        if(reset_stop || hwload_mode != 2)
            return 0;
        DELAY_LONG(kb);
    }
    // Give up
    ckb_err_fn("Too many send/recv failures. Dropping.\n", file, line);
    return 0;
}
```

Here is the call graph for this function:



5.38.3.4 int _usbsend (**usbdevice** ∗ *kb,* const **uchar** ∗ *messages,* int *count,* const char ∗ *file,* int *line*)

Parameters

	kb	THE usbdevice∗
	IN]	messages a Pointer to the first byte of the logical message
	IN]	count how many MSG_SIZE buffers is the logical message long?
	IN]	file for debugging
	IN]	line for debugging
in	*reset_stop*	global variable is read

Returns

number of Bytes sent (ideal == count ∗ MSG_SIZE);
0 if a block could not be sent and it was not a timeout OR **reset_stop** was required or **hwload_mode** is not set to "always"

_usbsend send a logical message completely to the given device

Todo A lot of different conditions are combined in this code. Don't think, it is good in every combination...

The main task of _usbsend () is to transfer the complete logical message from the buffer beginning with *messages* to **count** ∗ **MSG_SIZE**.

According to usb 2.0 specification, a USB transmits a maximum of 64 byte user data packets. For the transmission of longer messages we need a segmentation. And that is exactly what happens here.

The message is given one by one to os_usbsend() in MSG_SIZE (= 64) byte large bites.

Attention

This means that the buffer given as argument must be n ∗ MSG_SIZE Byte long.

An essential constant parameter which is relevant for os_usbsend() only is is_recv = 0, which means sending.

Now it gets a little complicated again:

- If os_usbsend() returns 0, only zero bytes could be sent in one of the packets, or it was an error (-1 from the systemcall), but not a timeout. How many Bytes were sent in total from earlier calls does not seem to matter, _usbsend() returns a total of 0.
- Returns os_usbsend() -1, first check if **reset_stop** is set globally or (incomprehensible) hwload_mode is not set to "always". In either case, _usbsend() returns 0, otherwise it is assumed to be a temporary transfer error and it simply retransmits the physical packet after a long delay.
- If the return value of os_usbsend() was neither 0 nor -1, it specifies the numer of bytes transferred.

  Here is an information hiding conflict with os_usbsend() (at least in the Linux version):

  If os_usbsend() can not transfer the entire packet, errors are thrown and the number of bytes sent is returned. _usbsend() interprets this as well and remembers the total number of bytes transferred in the local variable **total_sent**. Subsequently, however, transmission is continued with the next complete MSG_SIZE block and not with the first of the possibly missing bytes.

  Todo Check whether this is the same in the macOS variant. It is not dramatic, but if errors occur, it can certainly irritate the devices completely if they receive incomplete data streams. Do we have errors with the messages "Wrote YY bytes (expected 64)" in the system logs? If not, we do not need to look any further.

When the last packet is transferred, _usbsend() returns the effectively counted set of bytes (from **total_sent**). This at least gives the caller the opportunity to check whether something has been lost in the middle.

A bit strange is the structure of the program: Handling the **count** MSG_SIZE blocks to be transferred is done in the outer for (...) loop. Repeating the transfer with a treatable error is managed by the inner while(1) loop.

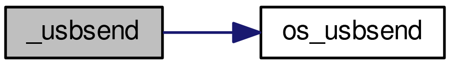
This must be considered when reading the code; The "break" on successful block transfer leaves the inner while, not the for (...).

Definition at line 532 of file usb.c.

References DELAY_LONG, DELAY_SHORT, hwload_mode, MSG_SIZE, os_usbsend(), and reset_stop.

```
532                                                                                          {
533     int total_sent = 0;
534     for(int i = 0; i < count; i++){
535         // Send each message via the OS function
536         while(1){
537             DELAY_SHORT(kb);
538             int res = os_usbsend(kb, messages + i * MSG_SIZE, 0, file, line);
539             if(res == 0)
540                 return 0;
541             else if(res != -1){
542                 total_sent += res;
543                 break;
544             }
545             // Stop immediately if the program is shutting down or hardware load is set to tryonce
546             if(reset_stop || hwload_mode != 2)
547                 return 0;
548             // Retry as long as the result is temporary failure
549             DELAY_LONG(kb);
550         }
551     }
552     return total_sent;
553 }
```

Here is the call graph for this function:



5.38.3.5 int closeusb (**usbdevice** ∗ *kb*)

Parameters

IN,OUT]	kb

Returns

Returns 0 (everytime. No error handling is done!)

closeusb Close a USB device and remove device entry.

An imutex lock ensures first of all, that no communication is currently running from the viewpoint of the driver to the user input device (ie the virtual driver with which characters or mouse movements are sent from the daemon to the operating system as inputs).

If the **kb** has an acceptable value = 0, the index of the device is looked for and with this index os_inputclose() is called. After this no more characters can be sent to the operating system.

Then the connection to the usb device is capped by os_closeusb().

Todo What is not yet comprehensible is the call to updateconnected() BEFORE os_closeusb(). Should that be in the other sequence? Or is updateconnected() not displaying the connected usb devices, but the representation which uinput devices are loaded? Questions about questions ...

If there is no valid **handle**, only updateconnected() is called. We are probably trying to disconnect a connection under construction. Not clear.

The cmd pipe as well as all open notify pipes are deleted via rmdevpath ().

This means that nothing can happen to the input path - so the device-specific imutex is unlocked again and remains unlocked.

Also the dmutex is unlocked now, but only to join the thread, which was originally taken under **kb->thread** (which started with _setupusb()) with pthread_join() again. Because of the closed devices that thread would have to quit sometime

See Also

the hack note with rmdevpath())

As soon as the thread is caught, the dmutex is locked again, which is what I do not understand yet: What other thread can do usb communication now?

If the vtabel exists for the given kb (why not? It seems to have race conditions here!!), via the vtable the actually device-specific, but still everywhere identical freeprofile() is called. This frees areas that are no longer needed. Then the **usbdevice** structure in its array is set to zero completely.

Error handling is rather unusual in closeusb(); Everything works (no matter what the called functions return), and closeusb() always returns zero (success).

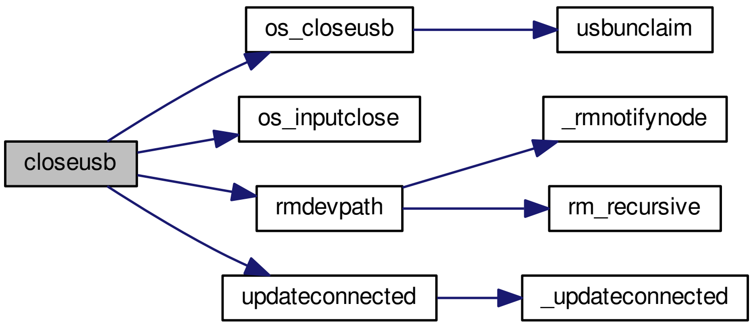
Definition at line 673 of file usb.c.

References ckb_info, devpath, dmutex, usbdevice::handle, imutex, INDEX_OF, keyboard, os_closeusb(), os_-inputclose(), rmdevpath(), usbdevice::thread, updateconnected(), and usbdevice::vtable.

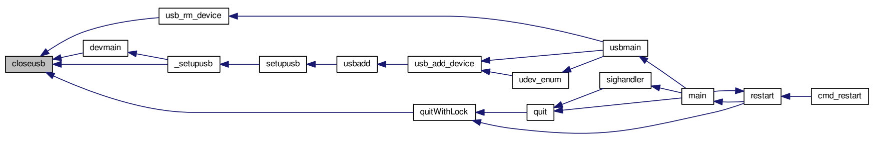
Referenced by _setupusb(), devmain(), quitWithLock(), and usb_rm_device().

```
673                         {
674     pthread_mutex_lock(imutex(kb));
675     if(kb->handle){
676         int index = INDEX_OF(kb, keyboard);
677         ckb_info("Disconnecting %s%d\n", devpath, index);
678         os_inputclose(kb);
679         updateconnected();
680         // Close USB device
681         os_closeusb(kb);
682     } else
683         updateconnected();
684     rmdevpath(kb);
685 
686     // Wait for thread to close
687     pthread_mutex_unlock(imutex(kb));
688     pthread_mutex_unlock(dmutex(kb));
689     pthread_join(kb->thread, 0);
690     pthread_mutex_lock(dmutex(kb));
691 
692     // Delete the profile and the control path
693     if(!kb->vtable)
694         return 0;
695     kb->vtable->freeprofile(kb);
696     memset(kb, 0, sizeof(usbdevice));
697     return 0;
698 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.38.3.6 void os_closeusb (usbdevice * *kb*)

Parameters

IN,OUT]	kb THE usbdevice∗

os_closeusb unclaim it, destroy the udev device and clear data structures at kb

os_closeusb is the linux specific implementation for closing an active usb port.

If a valid handle is given in the kb structure, the usb port is unclaimed (usbunclaim()).

The device in unrefenced via library function udev_device_unref().

handle, udev and the first char of kbsyspath are cleared to 0 (empty string for kbsyspath).

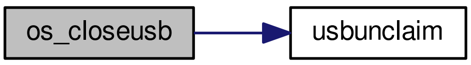
Definition at line 422 of file usb_linux.c.

References usbdevice::handle, INDEX_OF, kbsyspath, keyboard, usbdevice::udev, and usbunclaim().

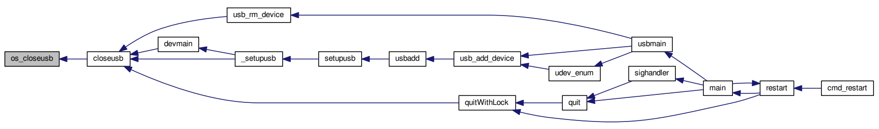
Referenced by closeusb().

```
                                   {
    if(kb->handle){
        usbunclaim(kb, 0);
        close(kb->handle - 1);
    }
    if(kb->udev)
        udev_device_unref(kb->udev);
    kb->handle = 0;
    kb->udev = 0;
    kbsyspath[INDEX_OF(kb, keyboard)][0] = 0;
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.38.3.7 void∗ os_inputmain (void ∗ *context*)

Parameters

context	THE usbdevice∗ ; Because os_inputmain() is started as a new thread, its formal parameter is named "context".

Returns

null

os_inputmain is run in a separate thread and will be detached from the main thread, so it needs to clean up its own resources.

Todo This function is a collection of many tasks. It should be divided into several sub-functions for the sake of greater convenience:

1. set up an URB (Userspace Ressource Buffer) to communicate with the USBDEVFS_∗ ioctl()s
2. perform the ioctl()
3. interpretate the information got into the URB buffer or handle error situations and retry operation or leave the endless loop
4. inform the os about the data
5. loop endless via 2.
6. if endless loop has gone, deinitalize the interface, free buffers etc.
7. return null

Here the actions in detail:

Monitor input transfers on all endpoints for non-RGB devices For RGB, monitor all but the last, as it's used for input/output

Get an usbdevfs_urb data structure and clear it via memset()

Hopefully the buffer lengths are equal for all devices with congruent types. You can find out the correctness for your device with lsusb –v or similar on macOS. Currently the following combinations are known and implemented:

device	detect with macro combination	endpoint #	buffer-length
each	none	0	8
RGB Mouse	IS_RGB && IS_MOUSE	1	10
RGB Keyboard	IS_RGB && !IS_MOUSE	1	21
RGB Mouse or Keyboard	IS_RGB	2	MSG_SIZE (64)
non RGB Mouse or Keyboard	!IS_RGB	1	4
non RGB Mouse or Keyboard	!IS_RGB	2	15

Now submit all the URBs via ioctl(USBDEVFS_SUBMITURB) with type USBDEVFS_URB_TYPE_INTERRUPT (the endpoints are defined as type interrupt). Endpoint number is 0x80..0x82 or 0x83, depending on the model.

The userSpaceFS knows the URBs now, so start monitoring input

if the ioctl returns something != 0, let's have a deeper look what happened. Broken devices or shutting down the entire system leads to closing the device and finishing this thread.

If just an EPIPE ocurred, give the device a CLEAR_HALT and resubmit the URB.

A correct REAPURB returns a Pointer to the URB which we now have a closer look into. Lock all following actions with imutex.

Process the input depending on type of device. Interprete the actual size of the URB buffer

device	detect with macro combination	seems to be endpoint #	actual buffer-length	function called

mouse (RGB and non RGB)	IS_MOUSE	nA	8, 10 or 11	hid_mouse_-translate()
mouse (RGB and non RGB)	IS_MOUSE	nA	MSG_SIZE (64)	corsair_-mousecopy()
RGB Keyboard	IS_RGB && !IS_MOUSE	1	8 (BIOS Mode)	hid_kb_translate()
RGB Keyboard	IS_RGB && !IS_MOUSE	2	5 or 21, KB inactive!	hid_kb_translate()
RGB Keyboard	IS_RGB && !IS_MOUSE	3?	MSG_SIZE	corsair_kbcopy()
non RGB Keyboard	!IS_RGB && !IS_MOUSE	nA	nA	hid_kb_translate()

The input data is transformed and copied to the kb structure. Now give it to the OS and unlock the imutex afterwards.

Re-submit the URB for the next run.

If the endless loop is terminated, clean up by discarding the URBs via ioctl(USBDEVFS_DISCARDURB), free the URB buffers and return a null pointer as thread exit code.

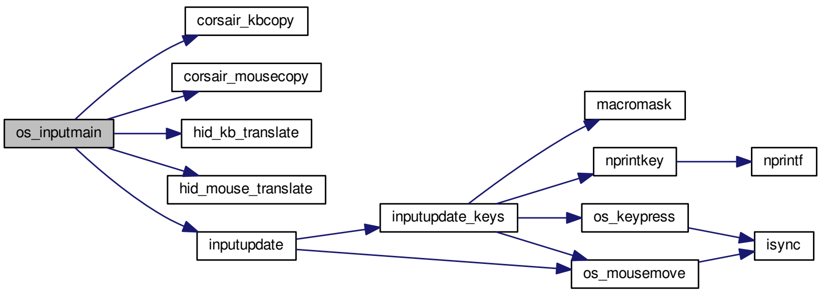
Definition at line 232 of file usb_linux.c.

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.

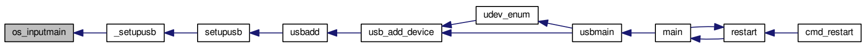
Referenced by _setupusb().

```
                                {
    usbdevice* kb = context;
    int fd = kb->handle - 1;
    short vendor = kb->vendor, product = kb->product;
    int index = INDEX_OF(kb, keyboard);
    ckb_info("Starting input thread for %s%d\n", devpath, index);

    int urbcount = IS_RGB(vendor, product) ? (kb->epcount - 1) : kb->
  epcount;

    struct usbdevfs_urb urbs[urbcount];
    memset(urbs, 0, sizeof(urbs));

    urbs[0].buffer_length = 8;
    if(IS_RGB(vendor, product)){
        if(IS_MOUSE(vendor, product))
            urbs[1].buffer_length = 10;
        else
            urbs[1].buffer_length = 21;
        urbs[2].buffer_length = MSG_SIZE;
        if(urbcount != 3)
            urbs[urbcount - 1].buffer_length = MSG_SIZE;
    } else {
        urbs[1].buffer_length = 4;
        urbs[2].buffer_length = 15;
    }

    for(int i = 0; i < urbcount; i++){
        urbs[i].type = USBDEVFS_URB_TYPE_INTERRUPT;
        urbs[i].endpoint = 0x80 | (i + 1);
        urbs[i].buffer = malloc(urbs[i].buffer_length);
        ioctl(fd, USBDEVFS_SUBMITURB, urbs + i);
    }

    while (1) {
        struct usbdevfs_urb* urb = 0;

        if (ioctl(fd, USBDEVFS_REAPURB, &urb)){
            if (errno == ENODEV || errno == ENOENT || errno == ESHUTDOWN)
                // Stop the thread if the handle closes
                break;
            else if(errno == EPIPE && urb){
                ioctl(fd, USBDEVFS_CLEAR_HALT, &urb->endpoint);
                // Re-submit the URB
                if(urb)
                    ioctl(fd, USBDEVFS_SUBMITURB, urb);
                urb = 0;
            }
        }

        if (urb) {
            pthread_mutex_lock(imutex(kb));
            if(IS_MOUSE(vendor, product)){
                switch(urb->actual_length){
                case 8:
                case 10:
                case 11:
                    // HID mouse input
                    hid_mouse_translate(kb->input.keys, &kb->input.rel_x, &kb->input.rel_y, -(urb->endpoint & 0xF), urb->actual_length, urb->buffer);
                    break;
                case MSG_SIZE:
                    // Corsair mouse input
                    corsair_mousecopy(kb->input.keys, -(urb->endpoint & 0xF), urb->buffer);
                    break;
                }
            } else if(IS_RGB(vendor, product)){
                switch(urb->actual_length){
                case 8:
                    // RGB EP 1: 6KRO (BIOS mode) input
                    hid_kb_translate(kb->input.keys, -1, urb->actual_length, urb->buffer);
                    break;
                case 21:
                case 5:
                    // RGB EP 2: NKRO (non-BIOS) input. Accept only if keyboard is inactive
                    if(!kb->active)
                        hid_kb_translate(kb->input.keys, -2, urb->actual_length, urb->buffer);
                    break;
                case MSG_SIZE:
                    // RGB EP 3: Corsair input
                    corsair_kbcopy(kb->input.keys, -(urb->endpoint & 0xF), urb->buffer);
                    break;
                }
            } else {
                // Non-RGB input
                hid_kb_translate(kb->input.keys, urb->endpoint & 0xF, urb->actual_length, urb->buffer);
            }
            inputupdate(kb);
            pthread_mutex_unlock(imutex(kb));
            ioctl(fd, USBDEVFS_SUBMITURB, urb);
            urb = 0;
        }
    }

    ckb_info("Stopping input thread for %s%d\n", devpath, index);
    for(int i = 0; i < urbcount; i++){
        ioctl(fd, USBDEVFS_DISCARDURB, urbs + i);
        free(urbs[i].buffer);
    }
    return 0;
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.38.3.8 int os_resetusb (**usbdevice** ∗ *kb,* const char ∗ *file,* int *line*)

Parameters

kb	THE usbdevice∗
file	filename for error messages
line	line where it is called for error messages

Returns

Returns 0 on success, -2 if device should be removed and -1 if reset should by tried again

os_resetusb is the os specific implementation for resetting usb

Try to reset an usb device in a linux user space driver.

1. unclaim the device, but do not reconnect the system driver (second param resetting = true)
2. reset the device via USBDEVFS_RESET command
3. claim the device again. Returns 0 on success, -2 if device should be removed and -1 if reset should by tried again

Todo it seems that no one wants to try the reset again. But I'v seen it somewhere...

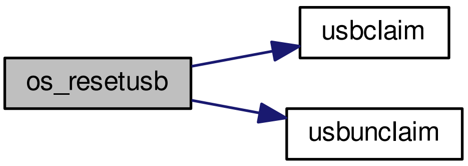
Definition at line 480 of file usb_linux.c.

References usbdevice::handle, TEST_RESET, usbclaim(), and usbunclaim().

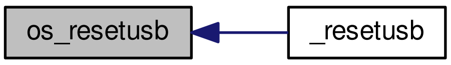
Referenced by _resetusb().

```
480                                                          {
481     TEST_RESET(usbunclaim(kb, 1));
482     TEST_RESET(ioctl(kb->handle - 1, USBDEVFS_RESET));
483     TEST_RESET(usbclaim(kb));
484     // Success!
485     return 0;
486 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.38.3.9 void os_sendindicators (usbdevice ∗ *kb*)

Parameters

kb	THE usbdevice∗

os_sendindicators update the indicators for the special keys (Numlock, Capslock and what else?)

os_sendindicators update the indicators for the special keys (Numlock, Capslock and what else?)

Read the data from kb->ileds ans send them via ioctl() to the keyboard.

bRequest-Type	bRequest	wValue	EP	size	Timeout	data
0x21	0x09	0x0200	Interface 0	MSG_SIZE 1 Byte	timeout 0,5ms	the message buffer pointer
Host to Device, Type=Class, Recipi-ent=Interface (why not endpoint?)	9 = SEND?	specific	0	1	500	struct∗ kb->ileds

The ioctl command is USBDEVFS_CONTROL.

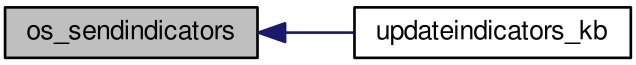
Definition at line 212 of file usb_linux.c.

References ckb_err, usbdevice::handle, and usbdevice::ileds.

Referenced by updateindicators_kb().

```
                                   {
    struct usbdevfs_ctrltransfer transfer = { 0x21, 0x09, 0x0200, 0x00, 1, 500, &kb->
  ileds };
    int res = ioctl(kb->handle - 1, USBDEVFS_CONTROL, &transfer);
    if(res <= 0)
        ckb_err("%s\n", res ? strerror(errno) : "No data written");
}
```

Here is the caller graph for this function:



5.38.3.10 int os_setupusb (usbdevice ∗ *kb*)

Parameters

kb	THE usbdevice∗

Returns

0 on success, -1 otherwise.

os_setupusb OS-specific setup for a specific usb device.

Perform the operating system-specific opening of the interface in os_setupusb(). As a result, some parameters should be set in kb (name, serial, fwversion, epcount = number of usb endpoints), and all endpoints should be claimed with usbclaim(). Claiming is the only point where os_setupusb() can produce an error (-1).

- Copy device description and serial
- Copy firmware version (needed to determine USB protocol)
- Do some output abaout connecting interfaces
- Claim the USB interfaces

Todo in these modules a pullrequest is outstanding

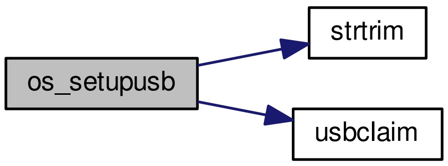
Definition at line 518 of file usb_linux.c.

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().

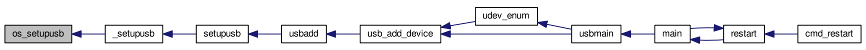
Referenced by _setupusb().

```
518                                {
521     struct udev_device* dev = kb->udev;
522     const char* name = udev_device_get_sysattr_value(dev, "product");
523     if(name)
524         strncpy(kb->name, name, KB_NAME_LEN);
525     strtrim(kb->name);
526     const char* serial = udev_device_get_sysattr_value(dev, "serial");
527     if(serial)
528         strncpy(kb->serial, serial, SERIAL_LEN);
529     strtrim(kb->serial);
532     const char* firmware = udev_device_get_sysattr_value(dev, "bcdDevice");
533     if(firmware)
534         sscanf(firmware, "%hx", &kb->fwversion);
535     else
536         kb->fwversion = 0;
537     int index = INDEX_OF(kb, keyboard);
540     ckb_info("Connecting %s at %s%d\n", kb->name, devpath, index);
541 
547     const char* ep_str = udev_device_get_sysattr_value(dev, "bNumInterfaces");
548     kb->epcount = 0;
549     if(ep_str)
550         sscanf(ep_str, "%d", &kb->epcount);
551     if(kb->epcount == 0){
552         // This shouldn't happen, but if it does, assume EP count based on what the device is supposed to
     have
553         kb->epcount = (HAS_FEATURES(kb, FEAT_RGB) ? 4 : 3);
554         ckb_warn("Unable to read endpoint count from udev, assuming %d...\n", kb->
   epcount);
555     }
556     if(usbclaim(kb)){
557         ckb_err("Failed to claim interfaces: %s\n", strerror(errno));
558         return -1;
559     }
560     return 0;
561 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.38.3.11 int os_usbrecv (**usbdevice** ∗ *kb,* **uchar** ∗ *in_msg,* const char ∗ *file,* int *line*)

Parameters

kb	THE usbdevice∗
in_msg	the buffer to fill with the message received
file	for debugging
line	for debugging

Returns

-1 on timeout, 0 on hard error, numer of bytes received otherwise

os_usbrecv does what its name says:

The comment at the beginning of the procedure causes the suspicion that the firmware versionspecific distinction is missing for receiving from usb endpoint 3 or 4. The commented code contains only the reception from EP4, but this may be wrong for a software version 2.0 or higher (see the code for os-usbsend ()).

So all the receiving is done via an ioctl() like in os_usbsend. The ioctl() is given a struct usbdevfs_ctrltransfer, in which the relevant parameters are entered:

bRequest-Type	bRequest	wValue	EP	size	Timeout	data
0xA1	0x01	0x0200	endpoint to be addressed from epcount - 1	MSG_SIZE	5ms	the message buffer pointer

Device to Host, Type=Class, Recipi-ent=Interface	1 = RECEIVE?	specific	Interface #	64	5000	in_msg

The ioctl() returns the number of bytes received. Here is the usual check again:

- If the return value is -1 AND the error is a timeout (ETIMEOUT), os_usbrecv() will return -1 to indicate that it is probably a recoverable problem and a retry is recommended.
- For another negative value or other error identifier OR 0 bytes are received, 0 is returned as an identifier for a heavy error.
- In all other cases, the function returns the number of bytes received.

If this is not the entire blocksize (MSG_SIZE bytes), an error message is issued on the standard error channel [warning "Read YY bytes (expected 64)"].

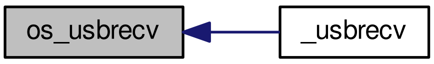
Definition at line 127 of file usb_linux.c.

References ckb_err_fn, ckb_warn_fn, usbdevice::epcount, usbdevice::handle, and MSG_SIZE.

Referenced by _usbrecv().

```
127                                                                          {
128     int res;
129     // This is what CUE does, but it doesn't seem to work on linux.
130     /*if(kb->fwversion >= 0x130){
131         struct usbdevfs_bulktransfer transfer;
132         memset(&transfer, 0, sizeof(transfer));
133         transfer.ep = 0x84;
134         transfer.len = MSG_SIZE;
135         transfer.timeout = 5000;
136         transfer.data = in_msg;
137         res = ioctl(kb->handle - 1, USBDEVFS_BULK, &transfer);
138     } else {*/
139         struct usbdevfs_ctrltransfer transfer = { 0xa1, 0x01, 0x0300, kb->
      epcount - 1, MSG_SIZE, 5000, in_msg };
140         res = ioctl(kb->handle - 1, USBDEVFS_CONTROL, &transfer);
141     //}
142     if(res <= 0){
143         ckb_err_fn("%s\n", file, line, res ? strerror(errno) : "No data read");
144         if(res == -1 && errno == ETIMEDOUT)
145             return -1;
146         else
147             return 0;
148     } else if(res != MSG_SIZE)
149         ckb_warn_fn("Read %d bytes (expected %d)\n", file, line, res,
      MSG_SIZE);
150 #ifdef DEBUG_USB_RECV
151     char converted[MSG_SIZE*3 + 1];
152     for(int i=0;i<MSG_SIZE;i++)
153         sprintf(&converted[i*3], "%02x ", in_msg[i]);
154     ckb_warn_fn("Recv %s\n", file, line, converted);
155 #endif
156     return res;
157 }
```

Here is the caller graph for this function:



5.38.3.12 int os_usbsend (usbdevice ∗ *kb,* const uchar ∗ *out_msg,* int *is_recv,* const char ∗ *file,* int *line*)

Parameters

kb	THE usbdevice∗
out_msg	the MSGSIZE char long buffer to send
is_recv	if true, just send an ioctl for further reading packets. If false, send the data at **out_msg**.
file	for debugging
line	for debugging

Returns

-1 on timeout (try again), 0 on hard error, numer of bytes sent otherwise

os_usbsend has two functions:

- if is_recv == false, it tries to send a given MSG_SIZE buffer via the usb interface given with kb.
- otherwise a request is sent via the usb device to initiate the receiving of a message from the remote device.

The functionality for sending distinguishes two cases, depending on the version number of the firmware of the connected device:

If the firmware is less or equal 1.2, the transmission is done via an ioctl(). The ioctl() is given a struct usbdevfs_-ctrltransfer, in which the relevant parameters are entered:

bRequest-Type	bRequest	wValue	EP	size	Timeout	data
0x21	0x09	0x0200	endpoint / IF to be addressed from epcount-1	MSG_SIZE	5000 (=5ms)	the message buffer pointer
Host to Device, Type=Class, Recipi-ent=Interface	9 = Send data?	specific	last or pre-last device #	64	5000	out_msg

The ioctl command is USBDEVFS_CONTROL.

The same constellation is used if the device is requested to send its data (is_recv = true).

For a more recent firmware and is_recv = false, the ioctl command USBDEVFS_CONTROL is not used (this tells the bus to enter the control mode), but the bulk method is used: USBDEVFS_BULK. This is astonishing, because all of the endpoints are type Interrupt, not bulk.

Anyhow, forthis purpose a different structure is used for the ioctl() (struct **usbdevfs_bulktransfer**) and this is also initialized differently:

The length and timeout parameters are given the same values as above. The formal parameter out_msg is also passed as a buffer pointer. For the endpoints, the firmware version is differentiated again:

For a firmware version between 1.3 and <2.0 endpoint 4 is used, otherwise (it can only be >=2.0) endpoint 3 is used.

Todo Since the handling of endpoints has already led to problems elsewhere, this implementation is extremely hardware-dependent and critical!

Eg. the new keyboard K95PLATINUMRGB has a version number significantly less than 2.0 - will it run with this implementation?

The ioctl() - no matter what type - returns the number of bytes sent. Now comes the usual check:

- If the return value is -1 AND the error is a timeout (ETIMEOUT), os_usbsend() will return -1 to indicate that it is probably a recoverable problem and a retry is recommended.

- For another negative value or other error identifier OR 0 bytes sent, 0 is returned as a heavy error identifier.
- In all other cases, the function returns the number of bytes sent.

If this is not the entire blocksize (MSG_SIZE bytes), an error message is issued on the standard error channel [warning "Wrote YY bytes (expected 64)"].

If DEBUG_USB is set during compilation, the number of bytes sent and their representation are logged to the error channel.

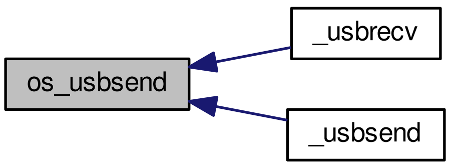
Definition at line 66 of file usb_linux.c.

References ckb_err_fn, ckb_warn_fn, usbdevice::epcount, usbdevice::fwversion, usbdevice::handle, and MSG_SIZE.

Referenced by _usbrecv(), and _usbsend().

```
66                                                                                      {
67     int res;
68     if(kb->fwversion >= 0x120 && !is_recv){
69         struct usbdevfs_bulktransfer transfer;
70         memset(&transfer, 0, sizeof(transfer));
71         transfer.ep = (kb->fwversion >= 0x130 && kb->fwversion < 0x200) ? 4 : 3;
72         transfer.len = MSG_SIZE;
73         transfer.timeout = 5000;
74         transfer.data = (void*)out_msg;
75         res = ioctl(kb->handle - 1, USBDEVFS_BULK, &transfer);
76     } else {
77         struct usbdevfs_ctrltransfer transfer = { 0x21, 0x09, 0x0200, kb->
       epcount - 1, MSG_SIZE, 5000, (void*)out_msg };
78         res = ioctl(kb->handle - 1, USBDEVFS_CONTROL, &transfer);
79     }
80     if(res <= 0){
81         ckb_err_fn("%s\n", file, line, res ? strerror(errno) : "No data written");
82         if(res == -1 && errno == ETIMEDOUT)
83             return -1;
84         else
85             return 0;
86     } else if(res != MSG_SIZE)
87         ckb_warn_fn("Wrote %d bytes (expected %d)\n", file, line, res,
       MSG_SIZE);
88 #ifdef DEBUG_USB
89     char converted[MSG_SIZE*3 + 1];
90     for(int i=0;i<MSG_SIZE;i++)
91         sprintf(&converted[i*3], "%02x ", out_msg[i]);
92     ckb_warn_fn("Sent %s\n", file, line, converted);
93 #endif
94     return res;
95 }
```

Here is the caller graph for this function:



5.38.3.13 const char∗ product_str (short *product*)

Parameters

product	is the *short* USB device product ID

Returns

string to identify a type of device (see below)

product_str returns a condensed view on what type of device we have.

At present, various models and their properties are known from corsair products. Some models differ in principle (mice and keyboards), others differ in the way they function (for example, RGB and non RGB), but they are very similar.

Here, only the first point is taken into consideration and we return a unified model string. If the model is not known with its number, *product_str* returns an empty string.

The model numbers and corresponding strings wwith the numbers in hex-string are defined in `usb.h`

At present, this function is used to initialize `kb->name` and to give information in debug strings.

Attention

The combinations below have to fit to the combinations in the macros mentioned above. So if you add a device with a new number, change both.

Todo There are macros defined in usb.h to detect all the combinations below. the only difference is the parameter: The macros need the *kb*∗, product_str() needs the *product ID*

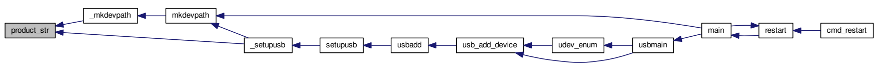
Definition at line 70 of file usb.c.

References P_HARPOON, 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_P-RO, P_STRAFE, and P_STRAFE_NRGB.

Referenced by _mkdevpath(), and _setupusb().

```
70                                      {
71     if(product == P_K95 || product == P_K95_NRGB || product ==
      P_K95_PLATINUM)
72         return "k95";
73     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)
74         return "k70";
75     if(product == P_K65 || product == P_K65_NRGB || product ==
      P_K65_LUX || product == P_K65_RFIRE)
76         return "k65";
77     if(product == P_STRAFE || product == P_STRAFE_NRGB)
78         return "strafe";
79     if(product == P_M65 || product == P_M65_PRO)
80         return "m65";
81     if(product == P_SABRE_O || product == P_SABRE_L || product ==
      P_SABRE_N || product == P_SABRE_O2 || product == P_HARPOON)
82         return "sabre";
83     if(product == P_SCIMITAR || product == P_SCIMITAR_PRO)
84         return "scimitar";
85     return "";
86 }
```

Here is the caller graph for this function:



5.38.3.14 int revertusb (usbdevice ∗ *kb*)

Parameters

kb	THE usbdevice∗

Returns

0 on success or if device needs firmware upgrade, -1 otherwise

revertusb sets a given device to inactive (hardware controlled) mode if not a fw-ugrade is indicated

First is checked, whether a firmware-upgrade is indicated for the device. If so, revertusb() returns 0.

Todo Why is this useful? Are there problems seen with deactivating a device with older fw-version??? Why isn't this an error indicating reason and we return success (0)?

Anyway, the following steps are similar to some other procs, dealing with low level usb handling:

- If we do not have an RGB device, a simple setting to Hardware-mode (NK95_HWON) is sent to the device via n95cmd().

  Todo The return value of nk95cmd() is ignored (but sending the ioctl may produce an error and _nk95_cmd will indicate this), instead revertusb() returns success in any case.

- If we have an RGB device, setactive() is called with second param active = false. That function will have a look on differences between keyboards and mice.

  More precisely setactive() is just a macro to call via the kb->vtable enties either the active() or the idle() function where the vtable points to. setactive() may return error indications. If so, revertusb() returns -1, otherwise 0 in any other case.

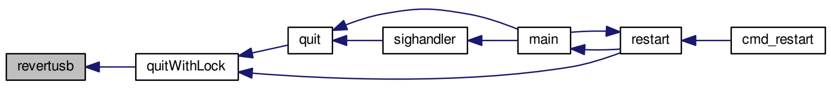
Definition at line 407 of file usb.c.

References FEAT_RGB, HAS_FEATURES, NEEDS_FW_UPDATE, NK95_HWON, nk95cmd, and setactive.

Referenced by quitWithLock().

```
                              {
    if(NEEDS_FW_UPDATE(kb))
        return 0;
    if(!HAS_FEATURES(kb, FEAT_RGB)){
        nk95cmd(kb, NK95_HWON);
        return 0;
    }
    if(setactive(kb, 0))
        return -1;
    return 0;
}
```

Here is the caller graph for this function:



5.38.3.15 void setupusb (usbdevice ∗ *kb*)

Attention

Lock a device's dmutex (see device.h) before accessing the USB interface.

Parameters

kb	THE usbdevice∗ used everywhere
OUT]	kb->thread is used to store the thread ID of the fresh created thread.

setupusb starts a thread with kb as parameter and _setupusb() as entrypoint.

Set up a USB device after its handle is open. Spawns a new thread _setupusb() with standard parameter kb. dmutex must be locked prior to calling this function. The function will unlock it when finished. In kb->thread the thread id is mentioned, because closeusb() needs this info for joining that thread again.

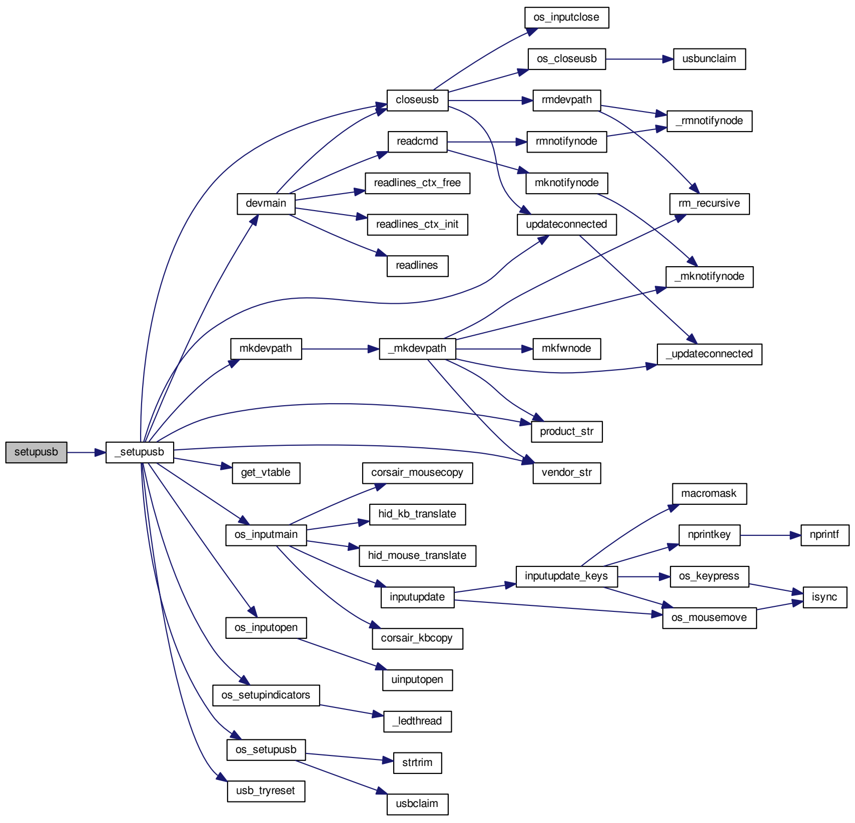
Definition at line 386 of file usb.c.

References _setupusb(), ckb_err, imutex, and usbdevice::thread.

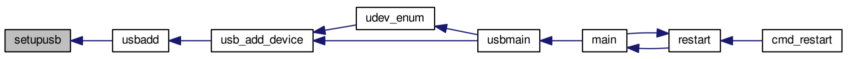
Referenced by usbadd().

```
386                          {
387     pthread_mutex_lock(imutex(kb));
388     if(pthread_create(&kb->thread, 0, _setupusb, kb))
389         ckb_err("Failed to create USB thread\n");
390 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.38.3.16 int usb_tryreset (usbdevice ∗ *kb*)

Parameters

in,out	*kb*	THE usbdevice∗
in	*reset_stop*	global variable is read

Returns

0 on success, -1 otherwise

usb_tryreset does what the name means: Try to reset the usb via resetusb()

This function is called if an usb command ran into an error in case of one of the following two situations:

- When setting up a new usb device and the start() function got an error (

  See Also

  _setupusb())

- If upgrading to a new firmware gets an error (

  See Also

  cmd_fwupdate()).

  The previous action which got the error will NOT be re-attempted.

In an endless loop usb_tryreset() tries to reset the given usb device via the macro resetusb().

This macro calls _resetusb() with debugging information.

_resetusb() sends a command via the operating system dependent function os_resetusb() and - if successful - reinitializes the device. os_resetusb() returns -2 to indicate a broken device and all structures should be removed for it.

In that case, the loop is terminated, an error message is produced and usb_tryreset() returns -1.

In case resetusb() has success, the endless loop is left via a return 0 (success).

If the return value from resetusb() is -1, the loop is continued with the next try.

If the global variable **reset_stop** is set directly when the function is called or after each try, usb_tryreset() stops working and returns -1.

Todo Why does usb_tryreset() hide the information returned from resetusb()? Isn't it needed by the callers?

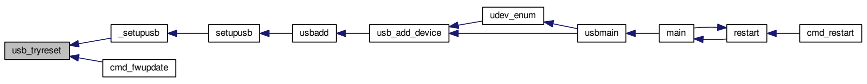
Definition at line 465 of file usb.c.

References ckb_err, ckb_info, reset_stop, and resetusb.

Referenced by _setupusb(), and cmd_fwupdate().

```
465                               {
466     if(reset_stop)
467         return -1;
468     ckb_info("Attempting reset...\n");
469     while(1){
470         int res = resetusb(kb);
471         if(!res){
472             ckb_info("Reset success\n");
473             return 0;
474         }
475         if(res == -2 || reset_stop)
476             break;
477     }
478     ckb_err("Reset failed. Disconnecting.\n");
479     return -1;
480 }
```

Here is the caller graph for this function:



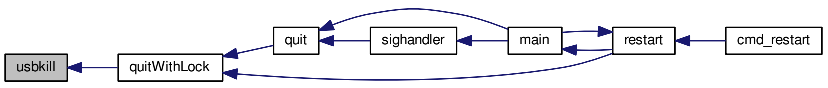
5.38.3.17 void usbkill ( )

Definition at line 803 of file usb_linux.c.

Referenced by quitWithLock().

```
803             {
804     udev_unref(udev);
805     udev = 0;
806 }
```

Here is the caller graph for this function:



5.38.3.18 int usbmain ( )

Start the USB main loop. Returns program exit code when finished.

usbmain is called by main() after setting up all other stuff.

Returns

0 normally or -1 if fatal error occurs (up to now only if no new devices are available)

First check whether the uinput module is loaded by the kernel.

Todo Why isn't missing of uinput a fatal error?

Create the udev object with udev_new() (is a function from libudev.h) terminate -1 if error

Enumerate all currently connected devices

Todo lae. here the work has to go on...

Definition at line 743 of file usb_linux.c.

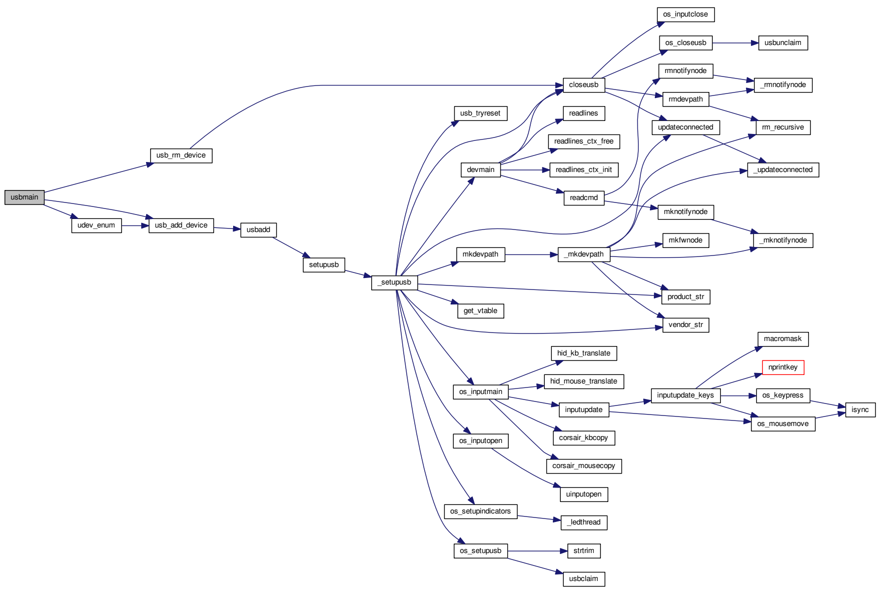
References ckb_fatal, ckb_warn, udev_enum(), usb_add_device(), and usb_rm_device().

Referenced by main().

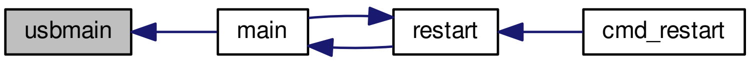
```
743            {
748     // Load the uinput module (if it's not loaded already)
749     if(system("modprobe uinput") != 0)
```

```
        ckb_warn("Failed to load uinput module\n");

    if(!(udev = udev_new())) {
        ckb_fatal("Failed to initialize udev in usbmain(), usb_linux.c\n");
        return -1;
    }

    udev_enum();

    // Done scanning. Enter a loop to poll for device updates
    struct udev_monitor* monitor = udev_monitor_new_from_netlink(udev, "udev");
    udev_monitor_filter_add_match_subsystem_devtype(monitor, "usb", 0);
    udev_monitor_enable_receiving(monitor);
    // Get an fd for the monitor
    int fd = udev_monitor_get_fd(monitor);
    fd_set fds;
    while(udev){
        FD_ZERO(&fds);
        FD_SET(fd, &fds);
        // Block until an event is read
        if(select(fd + 1, &fds, 0, 0, 0) > 0 && FD_ISSET(fd, &fds)){
            struct udev_device* dev = udev_monitor_receive_device(monitor);
            if(!dev)
                continue;
            const char* action = udev_device_get_action(dev);
            if(!action){
                udev_device_unref(dev);
                continue;
            }
            // Add/remove device
            if(!strcmp(action, "add")){
                int res = usb_add_device(dev);
                if(res == 0)
                    continue;
                // If the device matched but the handle wasn't opened correctly, re-enumerate (this sometimes solves the problem)
                if(res == -1)
                    udev_enum();
            } else if(!strcmp(action, "remove"))
                usb_rm_device(dev);
            udev_device_unref(dev);
        }
    }
    udev_monitor_unref(monitor);
    return 0;
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.38.3.19 const char∗ vendor_str (short *vendor*)

vendor_str Vendor/product string representations

Parameters

vendor	*short* vendor ID

Returns

a string: either "" or "corsair"

uncomment the following Define to see USB packets sent to the device

vendor_str returns "corsair" iff the given *vendor* argument is equal to *V_CORSAIR* (0x1bc) else it returns ""

Attention

There is also a string defined V_CORSAIR_STR, which returns the device number as string in hex "1b1c".

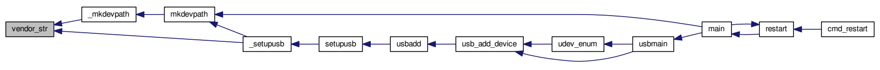
Definition at line 43 of file usb.c.

References V_CORSAIR.

Referenced by _mkdevpath(), and _setupusb().

```
43                                       {
44     if(vendor == V_CORSAIR)
45         return "corsair";
46     return "";
47 }
```

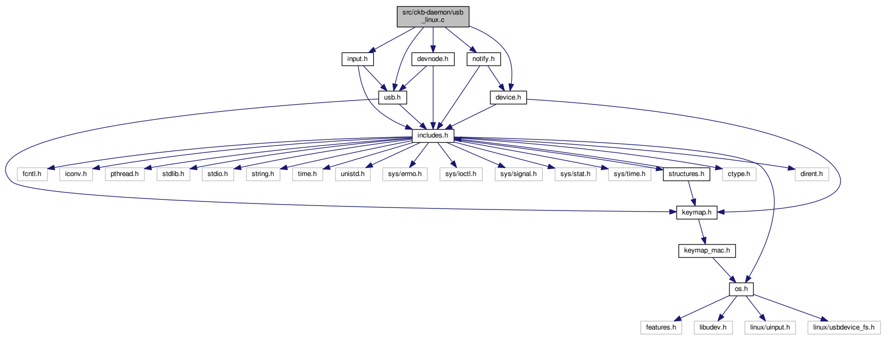
Here is the caller graph for this function:



5.39 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)

  TEST_RESET doesa "try / catch" for resetting the usb interface.
- #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)

  os_usbsend sends a data packet (MSG_SIZE = 64) Bytes long
- int os_usbrecv (usbdevice ∗kb, uchar ∗in_msg, const char ∗file, int line)

  os_usbrecv receives a max MSGSIZE long buffer from usb device
- int _nk95cmd (usbdevice ∗kb, uchar bRequest, ushort wValue, const char ∗file, int line)

  _nk95cmd If we control a non RGB keyboard, set the keyboard via ioctl with usbdevfs_ctrltransfer
- void os_sendindicators (usbdevice ∗kb)
- void ∗ os_inputmain (void ∗context)

  os_inputmain This function is run in a separate thread and will be detached from the main thread, so it needs to clean up its own resources.
- static int usbunclaim (usbdevice ∗kb, int resetting)
- void os_closeusb (usbdevice ∗kb)
- static 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)

  Add a udev device. Returns 0 if device was recognized/added.
- static void usb_rm_device (struct udev_device ∗dev)

  usb_rm_device find the usb port to remove and close it via closeusb().
- static void udev_enum ()

  udev_enum use the udev_enumerate_add_match_subsystem() to get all you need but only that.
- int usbmain ()
- void usbkill ()

  Stop the USB system.

Variables

- static char kbsyspath [9][FILENAME_MAX]

    all open usb devices have their system path names here in this array.
- static struct udev ∗ udev
- pthread_t usbthread

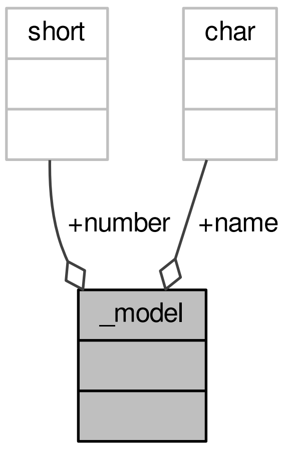
    struct udef is defined in /usr/include/libudev.h
- pthread_t udevthread
- static _model models []

5.39.1 Data Structure Documentation

5.39.1.1 struct _model

Definition at line 612 of file usb_linux.c.

Collaboration diagram for _model:



Data Fields

const char ∗	name	
short	number	

5.39.2 Macro Definition Documentation

5.39.2.1 #define N_MODELS (sizeof(models) / sizeof(_model))

Definition at line 650 of file usb_linux.c.

Referenced by usb_add_device().

5.39.2.2 #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 462 of file usb_linux.c.

Referenced by os_resetusb().

5.39.3 Function Documentation

5.39.3.1 int _nk95cmd (usbdevice * *kb,* uchar *bRequest,* ushort *wValue,* const char * *file,* int *line*)

To send control packets to a non RGB non color K95 Keyboard, use this function. Normally it is called via the nk95cmd() macro.

If it is the wrong device for which the function is called, 0 is returned and nothing done. Otherwise a usbdevfs_-ctrltransfer structure is filled and an USBDEVFS_CONTROL ioctl() called.

bRequest-Type	bRequest	wValue	EP	size	Timeout	data
0x40	see table below to switch hardware-modus at Keyboard	wValue	device	MSG_SIZE	5ms	the message buffer pointer
Host to Device, Type=Vendor, Recipi-ent=Device	bRequest parameter	given wValue Parameter	device 0	0 data to write	5000	null

If a 0 or a negative error number is returned by the ioctl, an error message is shown depending on the errno or "No data written" if retval was 0. In either case 1 is returned to indicate the error. If the ioctl returned a value $>$ 0, 0 is returned to indicate no error.

Currently the following combinations for bRequest and wValue are used:

Device	what it might to do	constant	bRequest	wValue
non RGB Keyboard	set HW-modus on (leave the ckb driver)	HWON	0x0002	0x0030
non RGB Keyboard	set HW-modus off (initialize the ckb driver)	HWOFF	0x0002	0x0001
non RGB Keyboard	set light modus M1 in single-color keyboards	NK95_M1	0x0014	0x0001
non RGB Keyboard	set light modus M2 in single-color keyboards	NK95_M2	0x0014	0x0002
non RGB Keyboard	set light modus M3 in single-color keyboards	NK95_M3	0x0014	0x0003

See Also

usb.h

Definition at line 187 of file usb_linux.c.

References ckb_err_fn, usbdevice::handle, P_K95_NRGB, and usbdevice::product.

```
                                                                                    {
    if(kb->product != P_K95_NRGB)
        return 0;
    struct usbdevfs_ctrltransfer transfer = { 0x40, bRequest, wValue, 0, 0, 5000, 0 };
    int res = ioctl(kb->handle - 1, USBDEVFS_CONTROL, &transfer);
    if(res <= 0){
        ckb_err_fn("%s\n", file, line, res ? strerror(errno) : "No data written");
        return 1;
    }
    return 0;
}
```

5.39.3.2 void os_closeusb (usbdevice * *kb*)

os_closeusb unclaim it, destroy the udev device and clear data structures at kb

os_closeusb is the linux specific implementation for closing an active usb port.

If a valid handle is given in the kb structure, the usb port is unclaimed (usbunclaim()).

The device in unrefenced via library function udev_device_unref().

handle, udev and the first char of kbsyspath are cleared to 0 (empty string for kbsyspath).

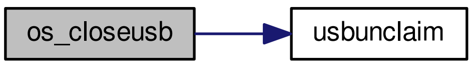
Definition at line 422 of file usb_linux.c.

References usbdevice::handle, INDEX_OF, kbsyspath, keyboard, usbdevice::udev, and usbunclaim().

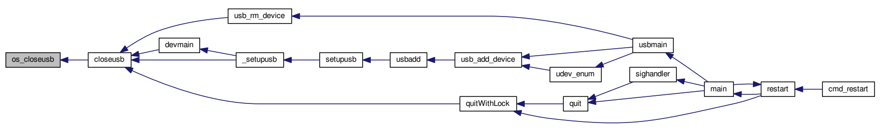
Referenced by closeusb().

```
                               {
    if(kb->handle){
        usbunclaim(kb, 0);
        close(kb->handle - 1);
    }
    if(kb->udev)
        udev_device_unref(kb->udev);
    kb->handle = 0;
    kb->udev = 0;
    kbsyspath[INDEX_OF(kb, keyboard)][0] = 0;
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.39.3.3 void∗ os_inputmain (void ∗ *context*)

os_inputmain is run in a separate thread and will be detached from the main thread, so it needs to clean up its own resources.

Todo This function is a collection of many tasks. It should be divided into several sub-functions for the sake of greater convenience:

1. set up an URB (Userspace Ressource Buffer) to communicate with the USBDEVFS_∗ ioctl()s
2. perform the ioctl()
3. interpretate the information got into the URB buffer or handle error situations and retry operation or leave the endless loop
4. inform the os about the data
5. loop endless via 2.
6. if endless loop has gone, deinitalize the interface, free buffers etc.
7. return null

Here the actions in detail:

Monitor input transfers on all endpoints for non-RGB devices For RGB, monitor all but the last, as it's used for input/output

Get an usbdevfs_urb data structure and clear it via memset()

Hopefully the buffer lengths are equal for all devices with congruent types. You can find out the correctness for your device with lsusb –v or similar on macOS. Currently the following combinations are known and implemented:

device	detect with macro combination	endpoint #	buffer-length
each	none	0	8
RGB Mouse	IS_RGB && IS_MOUSE	1	10
RGB Keyboard	IS_RGB && !IS_MOUSE	1	21
RGB Mouse or Keyboard	IS_RGB	2	MSG_SIZE (64)
non RGB Mouse or Keyboard	!IS_RGB	1	4
non RGB Mouse or Keyboard	!IS_RGB	2	15

Now submit all the URBs via ioctl(USBDEVFS_SUBMITURB) with type USBDEVFS_URB_TYPE_INTERRUPT (the endpoints are defined as type interrupt). Endpoint number is 0x80..0x82 or 0x83, depending on the model.

The userSpaceFS knows the URBs now, so start monitoring input

if the ioctl returns something != 0, let's have a deeper look what happened. Broken devices or shutting down the entire system leads to closing the device and finishing this thread.

If just an EPIPE ocurred, give the device a CLEAR_HALT and resubmit the URB.

A correct REAPURB returns a Pointer to the URB which we now have a closer look into. Lock all following actions with imutex.

Process the input depending on type of device. Interprete the actual size of the URB buffer

device	detect with macro combination	seems to be endpoint #	actual buffer-length	function called
mouse (RGB and non RGB)	IS_MOUSE	nA	8, 10 or 11	hid_mouse_-translate()
mouse (RGB and non RGB)	IS_MOUSE	nA	MSG_SIZE (64)	corsair_-mousecopy()
RGB Keyboard	IS_RGB && !IS_MOUSE	1	8 (BIOS Mode)	hid_kb_translate()
RGB Keyboard	IS_RGB && !IS_MOUSE	2	5 or 21, KB inactive!	hid_kb_translate()
RGB Keyboard	IS_RGB && !IS_MOUSE	3?	MSG_SIZE	corsair_kbcopy()
non RGB Keyboard	!IS_RGB && !IS_MOUSE	nA	nA	hid_kb_translate()

The input data is transformed and copied to the kb structure. Now give it to the OS and unlock the imutex afterwards.

Re-submit the URB for the next run.

If the endless loop is terminated, clean up by discarding the URBs via ioctl(USBDEVFS_DISCARDURB), free the URB buffers and return a null pointer as thread exit code.

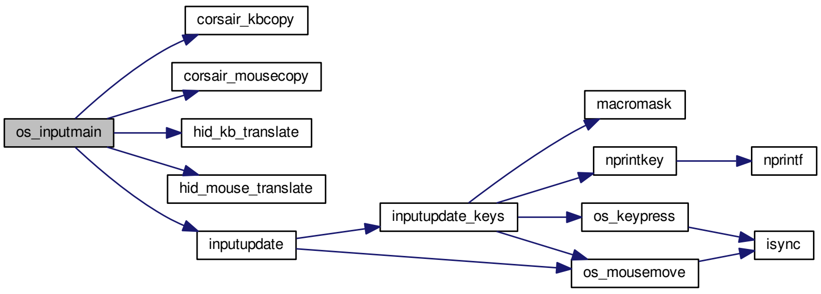
Definition at line 232 of file usb_linux.c.

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.

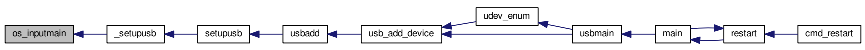
Referenced by _setupusb().

```
                                {
    usbdevice* kb = context;
    int fd = kb->handle - 1;
    short vendor = kb->vendor, product = kb->product;
    int index = INDEX_OF(kb, keyboard);
    ckb_info("Starting input thread for %s%d\n", devpath, index);

    int urbcount = IS_RGB(vendor, product) ? (kb->epcount - 1) : kb->
  epcount;

    struct usbdevfs_urb urbs[urbcount];
    memset(urbs, 0, sizeof(urbs));

    urbs[0].buffer_length = 8;
    if(IS_RGB(vendor, product)){
        if(IS_MOUSE(vendor, product))
            urbs[1].buffer_length = 10;
        else
            urbs[1].buffer_length = 21;
        urbs[2].buffer_length = MSG_SIZE;
        if(urbcount != 3)
            urbs[urbcount - 1].buffer_length = MSG_SIZE;
    } else {
        urbs[1].buffer_length = 4;
        urbs[2].buffer_length = 15;
    }

    for(int i = 0; i < urbcount; i++){
        urbs[i].type = USBDEVFS_URB_TYPE_INTERRUPT;
        urbs[i].endpoint = 0x80 | (i + 1);
        urbs[i].buffer = malloc(urbs[i].buffer_length);
        ioctl(fd, USBDEVFS_SUBMITURB, urbs + i);
    }

    while (1) {
        struct usbdevfs_urb* urb = 0;

        if (ioctl(fd, USBDEVFS_REAPURB, &urb)){
            if (errno == ENODEV || errno == ENOENT || errno == ESHUTDOWN)
                // Stop the thread if the handle closes
                break;
            else if(errno == EPIPE && urb){
                ioctl(fd, USBDEVFS_CLEAR_HALT, &urb->endpoint);
                // Re-submit the URB
                if(urb)
                    ioctl(fd, USBDEVFS_SUBMITURB, urb);
                urb = 0;
            }
        }

        if (urb) {
            pthread_mutex_lock(imutex(kb));
            if(IS_MOUSE(vendor, product)){
                switch(urb->actual_length){
                case 8:
                case 10:
                case 11:
                    // HID mouse input
                    hid_mouse_translate(kb->input.keys, &kb->input.rel_x, &kb->input.rel_y, -(urb->endpoint & 0xF), urb->actual_length, urb->buffer);
                    break;
                case MSG_SIZE:
                    // Corsair mouse input
                    corsair_mousecopy(kb->input.keys, -(urb->endpoint & 0xF), urb->buffer);
                    break;
                }
            } else if(IS_RGB(vendor, product)){
                switch(urb->actual_length){
                case 8:
                    // RGB EP 1: 6KRO (BIOS mode) input
                    hid_kb_translate(kb->input.keys, -1, urb->actual_length, urb->buffer);
                    break;
                case 21:
                case 5:
                    // RGB EP 2: NKRO (non-BIOS) input. Accept only if keyboard is inactive
                    if(!kb->active)
                        hid_kb_translate(kb->input.keys, -2, urb->actual_length, urb->buffer);
                    break;
                case MSG_SIZE:
                    // RGB EP 3: Corsair input
                    corsair_kbcopy(kb->input.keys, -(urb->endpoint & 0xF), urb->buffer);
                    break;
                }
            } else {
                // Non-RGB input
                hid_kb_translate(kb->input.keys, urb->endpoint & 0xF, urb->actual_length, urb->buffer);
            }
            inputupdate(kb);
            pthread_mutex_unlock(imutex(kb));
            ioctl(fd, USBDEVFS_SUBMITURB, urb);
            urb = 0;
        }
    }

    ckb_info("Stopping input thread for %s%d\n", devpath, index);
    for(int i = 0; i < urbcount; i++){
        ioctl(fd, USBDEVFS_DISCARDURB, urbs + i);
        free(urbs[i].buffer);
    }
    return 0;
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.39.3.4 int os_resetusb (**usbdevice** ∗ *kb,* const char ∗ *file,* int *line*)

os_resetusb is the os specific implementation for resetting usb

Try to reset an usb device in a linux user space driver.

1. unclaim the device, but do not reconnect the system driver (second param resetting = true)
2. reset the device via USBDEVFS_RESET command
3. claim the device again. Returns 0 on success, -2 if device should be removed and -1 if reset should by tried again

Todo it seems that no one wants to try the reset again. But I'v seen it somewhere...

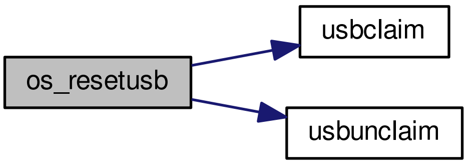
Definition at line 480 of file usb_linux.c.

References usbdevice::handle, TEST_RESET, usbclaim(), and usbunclaim().

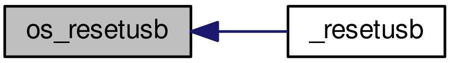
Referenced by _resetusb().

```
480                                                         {
481     TEST_RESET(usbunclaim(kb, 1));
482     TEST_RESET(ioctl(kb->handle - 1, USBDEVFS_RESET));
483     TEST_RESET(usbclaim(kb));
484     // Success!
485     return 0;
486 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.39.3.5 void os_sendindicators (**usbdevice** ∗ *kb*)

os_sendindicators update the indicators for the special keys (Numlock, Capslock and what else?)

os_sendindicators update the indicators for the special keys (Numlock, Capslock and what else?)

Read the data from kb->ileds ans send them via ioctl() to the keyboard.

bRequest-Type	**bRequest**	**wValue**	**EP**	**size**	**Timeout**	**data**
0x21	0x09	0x0200	Interface 0	MSG_SIZE 1 Byte	timeout 0,5ms	the message buffer pointer
Host to Device, Type=Class, Recipi-ent=Interface (why not endpoint?)	9 = SEND?	specific	0	1	500	struct∗ kb->ileds

The ioctl command is USBDEVFS_CONTROL.

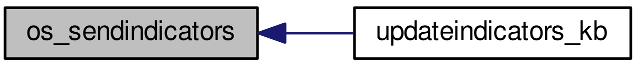
Definition at line 212 of file usb_linux.c.

References ckb_err, usbdevice::handle, and usbdevice::ileds.

Referenced by updateindicators_kb().

```
                                     {
    struct usbdevfs_ctrltransfer transfer = { 0x21, 0x09, 0x0200, 0x00, 1, 500, &kb->
  ileds };
    int res = ioctl(kb->handle - 1, USBDEVFS_CONTROL, &transfer);
    if(res <= 0)
        ckb_err("%s\n", res ? strerror(errno) : "No data written");
}
```

Here is the caller graph for this function:



5.39.3.6 int os_setupusb (usbdevice ∗ *kb*)

os_setupusb OS-specific setup for a specific usb device.

Perform the operating system-specific opening of the interface in os_setupusb(). As a result, some parameters should be set in kb (name, serial, fwversion, epcount = number of usb endpoints), and all endpoints should be claimed with usbclaim(). Claiming is the only point where os_setupusb() can produce an error (-1).

- Copy device description and serial
- Copy firmware version (needed to determine USB protocol)
- Do some output abaout connecting interfaces
- Claim the USB interfaces

Todo in these modules a pullrequest is outstanding

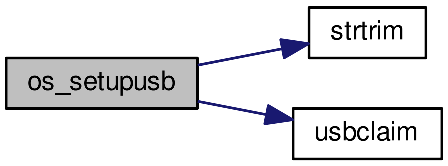
Definition at line 518 of file usb_linux.c.

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().

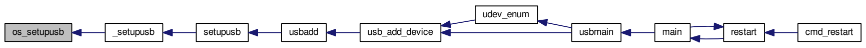
Referenced by _setupusb().

```
                                   {
    struct udev_device* dev = kb->udev;
    const char* name = udev_device_get_sysattr_value(dev, "product");
    if(name)
        strncpy(kb->name, name, KB_NAME_LEN);
    strtrim(kb->name);
    const char* serial = udev_device_get_sysattr_value(dev, "serial");
    if(serial)
        strncpy(kb->serial, serial, SERIAL_LEN);
    strtrim(kb->serial);
    const char* firmware = udev_device_get_sysattr_value(dev, "bcdDevice");
    if(firmware)
        sscanf(firmware, "%hx", &kb->fwversion);
    else
        kb->fwversion = 0;
    int index = INDEX_OF(kb, keyboard);
    ckb_info("Connecting %s at %s%d\n", kb->name, devpath, index);

    const char* ep_str = udev_device_get_sysattr_value(dev, "bNumInterfaces");
    kb->epcount = 0;
    if(ep_str)
        sscanf(ep_str, "%d", &kb->epcount);
    if(kb->epcount == 0){
        // This shouldn't happen, but if it does, assume EP count based on what the device is supposed to
   have
        kb->epcount = (HAS_FEATURES(kb, FEAT_RGB) ? 4 : 3);
        ckb_warn("Unable to read endpoint count from udev, assuming %d...\n", kb->
  epcount);
    }
    if(usbclaim(kb)){
        ckb_err("Failed to claim interfaces: %s\n", strerror(errno));
        return -1;
    }
    return 0;
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.39.3.7 int os_usbrecv (usbdevice * *kb,* uchar * *in_msg,* const char * *file,* int *line*)

os_usbrecv does what its name says:

The comment at the beginning of the procedure causes the suspicion that the firmware versionspecific distinction is missing for receiving from usb endpoint 3 or 4. The commented code contains only the reception from EP4, but this may be wrong for a software version 2.0 or higher (see the code for os-usbsend ()).

So all the receiving is done via an ioctl() like in os_usbsend. The ioctl() is given a struct usbdevfs_ctrltransfer, in which the relevant parameters are entered:

bRequest-Type	bRequest	wValue	EP	size	Timeout	data
0xA1	0x01	0x0200	endpoint to be addressed from epcount - 1	MSG_SIZE	5ms	the message buffer pointer
Device to Host, Type=Class, Recipi-ent=Interface	1 = RECEIVE?	specific	Interface #	64	5000	in_msg

The ioctl() returns the number of bytes received. Here is the usual check again:

- If the return value is -1 AND the error is a timeout (ETIMEOUT), os_usbrecv() will return -1 to indicate that it is probably a recoverable problem and a retry is recommended.
- For another negative value or other error identifier OR 0 bytes are received, 0 is returned as an identifier for a heavy error.

- In all other cases, the function returns the number of bytes received.

If this is not the entire blocksize (MSG_SIZE bytes), an error message is issued on the standard error channel [warning "Read YY bytes (expected 64)"].

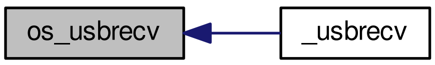
Definition at line 127 of file usb_linux.c.

References ckb_err_fn, ckb_warn_fn, usbdevice::epcount, usbdevice::handle, and MSG_SIZE.

Referenced by _usbrecv().

```
                                                                            {
    int res;
    // This is what CUE does, but it doesn't seem to work on linux.
    /*if(kb->fwversion >= 0x130){
        struct usbdevfs_bulktransfer transfer;
        memset(&transfer, 0, sizeof(transfer));
        transfer.ep = 0x84;
        transfer.len = MSG_SIZE;
        transfer.timeout = 5000;
        transfer.data = in_msg;
        res = ioctl(kb->handle - 1, USBDEVFS_BULK, &transfer);
    } else {*/
        struct usbdevfs_ctrltransfer transfer = { 0xa1, 0x01, 0x0300, kb->
  epcount - 1, MSG_SIZE, 5000, in_msg };
        res = ioctl(kb->handle - 1, USBDEVFS_CONTROL, &transfer);
    //}
    if(res <= 0){
        ckb_err_fn("%s\n", file, line, res ? strerror(errno) : "No data read");
        if(res == -1 && errno == ETIMEDOUT)
            return -1;
        else
            return 0;
    } else if(res != MSG_SIZE)
        ckb_warn_fn("Read %d bytes (expected %d)\n", file, line, res,
  MSG_SIZE);
#ifdef DEBUG_USB_RECV
    char converted[MSG_SIZE*3 + 1];
    for(int i=0;i<MSG_SIZE;i++)
        sprintf(&converted[i*3], "%02x ", in_msg[i]);
    ckb_warn_fn("Recv %s\n", file, line, converted);
#endif
    return res;
}
```

Here is the caller graph for this function:



5.39.3.8 int os_usbsend (usbdevice * *kb,* const uchar * *out_msg,* int *is_recv,* const char * *file,* int *line*)

os_usbsend has two functions:

- if is_recv == false, it tries to send a given MSG_SIZE buffer via the usb interface given with kb.
- otherwise a request is sent via the usb device to initiate the receiving of a message from the remote device.

The functionality for sending distinguishes two cases, depending on the version number of the firmware of the connected device:

If the firmware is less or equal 1.2, the transmission is done via an ioctl(). The ioctl() is given a struct usbdevfs_-ctrltransfer, in which the relevant parameters are entered:

bRequest-Type	bRequest	wValue	EP	size	Timeout	data
0x21	0x09	0x0200	endpoint / IF to be addressed from epcount-1	MSG_SIZE	5000 (=5ms)	the message buffer pointer
Host to Device, Type=Class, Recipi-ent=Interface	9 = Send data?	specific	last or pre-last device #	64	5000	out_msg

The ioctl command is USBDEVFS_CONTROL.

The same constellation is used if the device is requested to send its data (is_recv = true).

For a more recent firmware and is_recv = false, the ioctl command USBDEVFS_CONTROL is not used (this tells the bus to enter the control mode), but the bulk method is used: USBDEVFS_BULK. This is astonishing, because all of the endpoints are type Interrupt, not bulk.

Anyhow, forthis purpose a different structure is used for the ioctl() (struct **usbdevfs_bulktransfer**) and this is also initialized differently:

The length and timeout parameters are given the same values as above. The formal parameter out_msg is also passed as a buffer pointer. For the endpoints, the firmware version is differentiated again:

For a firmware version between 1.3 and <2.0 endpoint 4 is used, otherwise (it can only be >=2.0) endpoint 3 is used.

Todo Since the handling of endpoints has already led to problems elsewhere, this implementation is extremely hardware-dependent and critical!

Eg. the new keyboard K95PLATINUMRGB has a version number significantly less than 2.0 - will it run with this implementation?

The ioctl() - no matter what type - returns the number of bytes sent. Now comes the usual check:

- If the return value is -1 AND the error is a timeout (ETIMEOUT), os_usbsend() will return -1 to indicate that it is probably a recoverable problem and a retry is recommended.
- For another negative value or other error identifier OR 0 bytes sent, 0 is returned as a heavy error identifier.
- In all other cases, the function returns the number of bytes sent.

If this is not the entire blocksize (MSG_SIZE bytes), an error message is issued on the standard error channel [warning "Wrote YY bytes (expected 64)"].

If DEBUG_USB is set during compilation, the number of bytes sent and their representation are logged to the error channel.

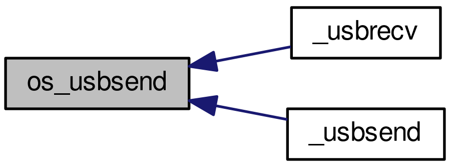
Definition at line 66 of file usb_linux.c.

References ckb_err_fn, ckb_warn_fn, usbdevice::epcount, usbdevice::fwversion, usbdevice::handle, and MSG_SI-ZE.

Referenced by _usbrecv(), and _usbsend().

```
                                                                                    {
    int res;
    if(kb->fwversion >= 0x120 && !is_recv){
        struct usbdevfs_bulktransfer transfer;
        memset(&transfer, 0, sizeof(transfer));
        transfer.ep = (kb->fwversion >= 0x130 && kb->fwversion < 0x200) ? 4 : 3;
        transfer.len = MSG_SIZE;
        transfer.timeout = 5000;
        transfer.data = (void*)out_msg;
        res = ioctl(kb->handle - 1, USBDEVFS_BULK, &transfer);
    } else {
        struct usbdevfs_ctrltransfer transfer = { 0x21, 0x09, 0x0200, kb->
    epcount - 1, MSG_SIZE, 5000, (void*)out_msg };
        res = ioctl(kb->handle - 1, USBDEVFS_CONTROL, &transfer);
    }
    if(res <= 0){
        ckb_err_fn("%s\n", file, line, res ? strerror(errno) : "No data written");
        if(res == -1 && errno == ETIMEDOUT)
            return -1;
        else
            return 0;
    } else if(res != MSG_SIZE)
        ckb_warn_fn("Wrote %d bytes (expected %d)\n", file, line, res,
    MSG_SIZE);
#ifdef DEBUG_USB
    char converted[MSG_SIZE*3 + 1];
    for(int i=0;i<MSG_SIZE;i++)
        sprintf(&converted[i*3], "%02x ", out_msg[i]);
    ckb_warn_fn("Sent %s\n", file, line, converted);
#endif
    return res;
}
```

Here is the caller graph for this function:



5.39.3.9 void strtrim (char * *string*)

strtrim trims a string by removing leading and trailing spaces.

Parameters

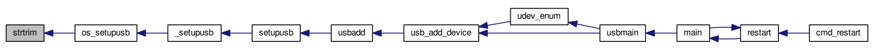
string	

Definition at line 493 of file usb_linux.c.

Referenced by os_setupusb().

```
                            {
    // Find last non-space
    char* last = string;
    for(char* c = string; *c != 0; c++){
        if(!isspace(*c))
            last = c;
    }
    last[1] = 0;
    // Find first non-space
    char* first = string;
    for(; *first != 0; first++){
        if(!isspace(*first))
            break;
    }
    if(first != string)
        memmove(string, first, last - first);
}
```

Here is the caller graph for this function:



5.39.3.10 static void udev_enum ( ) [static]

Reduce the hits of the enumeration by limiting to usb as technology and corsair as idVendor. Then filter with udev_enumerate_scan_devices () all hits.

The following call to udev_enumerate_get_list_entry() fetches the entire hitlist as udev_list_entry ∗.

Use udev_list_entry_foreach() to iterate through the hit set.

If both the device name exists (udev_list_entry_get_name) and the subsequent creation of a new udev_device (udev_device_new_from_syspath) is ok, the new device is added to the list with usb_add_device().

If the latter does not work, the new device is released again (udev_device_unref ()).

After the last iteration, the enumerator is released with udev_enumerate_unref ().

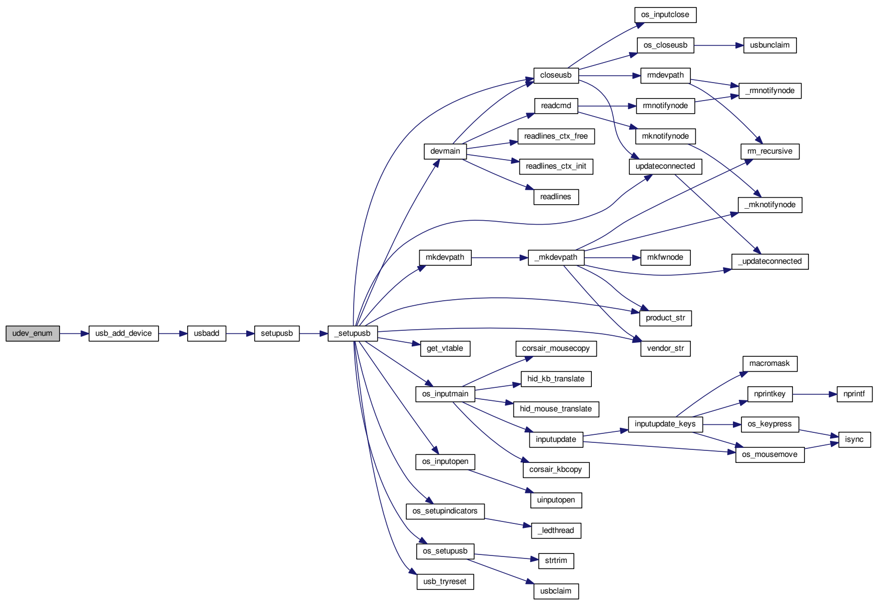
Definition at line 715 of file usb_linux.c.

References usb_add_device(), and V_CORSAIR_STR.

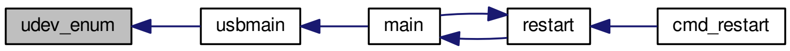
Referenced by usbmain().

```
715                         {
716     struct udev_enumerate* enumerator = udev_enumerate_new(udev);
717     udev_enumerate_add_match_subsystem(enumerator, "usb");
718     udev_enumerate_add_match_sysattr(enumerator, "idVendor", V_CORSAIR_STR);
719     udev_enumerate_scan_devices(enumerator);
720     struct udev_list_entry* devices, *dev_list_entry;
721     devices = udev_enumerate_get_list_entry(enumerator);
722 
723     udev_list_entry_foreach(dev_list_entry, devices){
724         const char* path = udev_list_entry_get_name(dev_list_entry);
725         if(!path)
726             continue;
727         struct udev_device* dev = udev_device_new_from_syspath(udev, path);
728         if(!dev)
729             continue;
730         // If the device matches a recognized device ID, open it
731         if(usb_add_device(dev))
732             // Release device if not
733             udev_device_unref(dev);
734     }
735     udev_enumerate_unref(enumerator);
736 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.39.3.11 static int usb_add_device (struct udev_device ∗ *dev*) [static]

If the device id can be found, call usbadd() with the appropriate parameters.

Parameters

dev	the functions usb_∗_device get a struct udev∗ with the neccessary hardware-related information.

Returns

the retval of usbadd() or 1 if either vendor is not corsair or product is not mentioned in model[].

First get the idVendor via udev_device_get_sysattr_value(). If this is equal to the ID-string of corsair ("1b1c"), get the idProduct on the same way.

If we can find the model name in the model array, call usbadd() with the model number.

Todo So why the hell not a transformation between the string and the short presentation? Lets check if the string representation is used elsewhere.

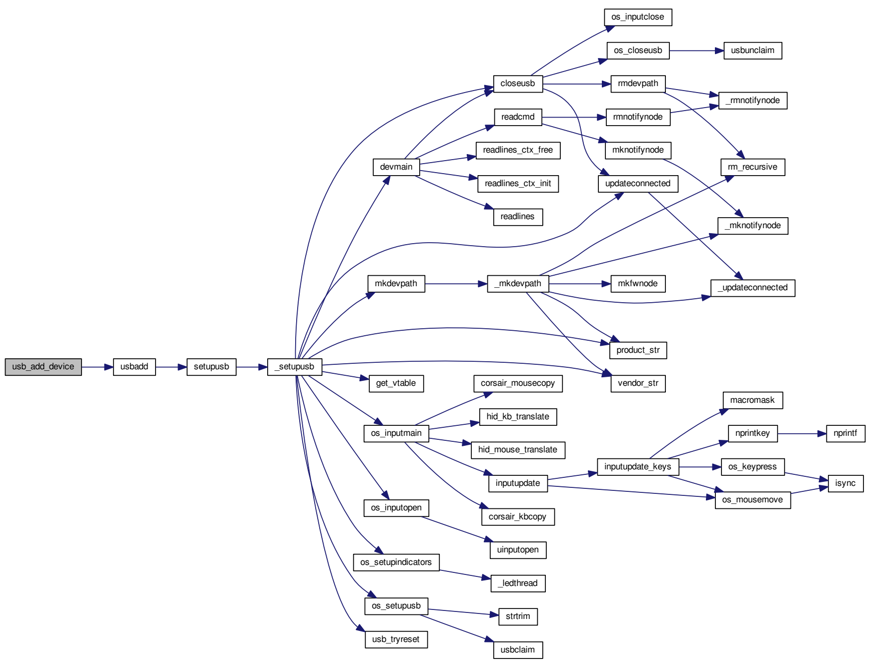
Definition at line 663 of file usb_linux.c.

References N_MODELS, usbadd(), V_CORSAIR, and V_CORSAIR_STR.

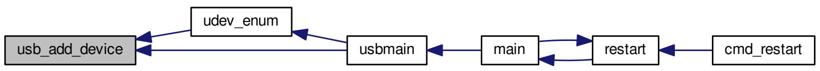
Referenced by udev_enum(), and usbmain().

```
                                                   {
    const char* vendor = udev_device_get_sysattr_value(dev, "idVendor");
    if(vendor && !strcmp(vendor, V_CORSAIR_STR)){
        const char* product = udev_device_get_sysattr_value(dev, "idProduct");
        if(product){
            for(_model* model = models; model < models +
      N_MODELS; model++){
                if(!strcmp(product, model->name)){
                    return usbadd(dev, V_CORSAIR, model->number);
                }
            }
        }
    }
    return 1;
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.39.3.12 static void usb_rm_device (struct udev_device ∗ *dev*) [static]

Parameters

dev	the functions usb_∗_device get a struct udev∗ with the neccessary hardware-related information.

First try to find the system path of the device given in parameter dev. The index where the name is found is the same index we need to address the global keyboard array. That array holds all usbdevices.

Searching for the correct name in kbsyspath-array and closing the usb via closeusb() are protected by lock..unlock of the corresponding devmutex arraymember.

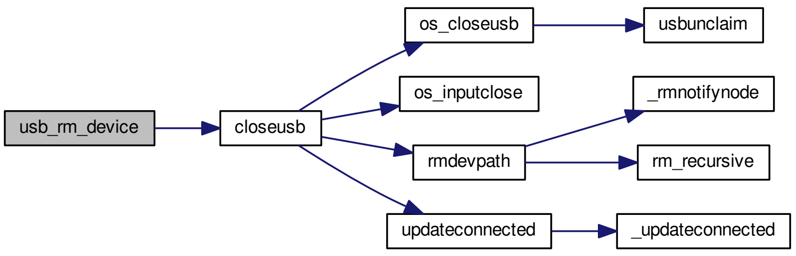
Definition at line 688 of file usb_linux.c.

References closeusb(), DEV_MAX, devmutex, kbsyspath, and keyboard.

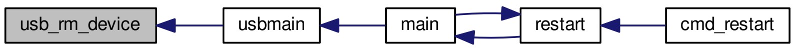
Referenced by usbmain().

```
                                                  {
    // Device removed. Look for it in our list of keyboards
    const char* syspath = udev_device_get_syspath(dev);
    if(!syspath || syspath[0] == 0)
        return;
    for(int i = 1; i < DEV_MAX; i++){
        pthread_mutex_lock(devmutex + i);
        if(!strcmp(syspath, kbsyspath[i]))
            closeusb(keyboard + i);
        pthread_mutex_unlock(devmutex + i);
    }
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.39.3.13 int usbadd (struct udev_device ∗ *dev,* short *vendor,* short *product*)

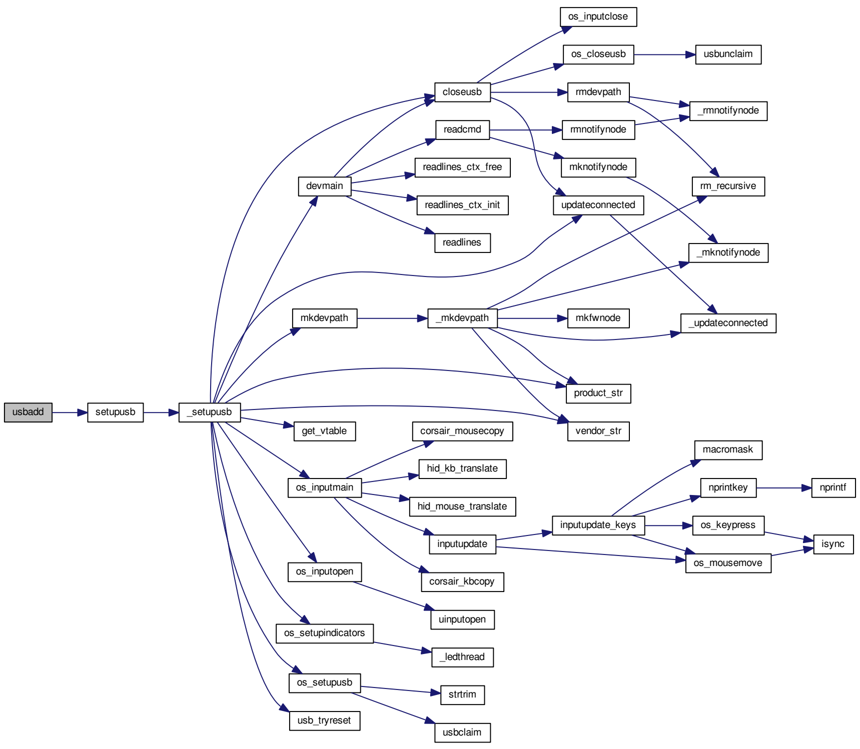
Definition at line 563 of file usb_linux.c.

References ckb_err, DEV_MAX, dmutex, usbdevice::handle, IS_CONNECTED, kbsyspath, keyboard, usbdevice::product, setupusb(), usbdevice::udev, and usbdevice::vendor.

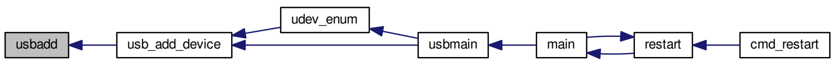
Referenced by usb_add_device().

```
                                                                {
    const char* path = udev_device_get_devnode(dev);
    const char* syspath = udev_device_get_syspath(dev);
    if(!path || !syspath || path[0] == 0 || syspath[0] == 0){
        ckb_err("Failed to get device path\n");
        return -1;
    }
    // Find a free USB slot
    for(int index = 1; index < DEV_MAX; index++){
        usbdevice* kb = keyboard + index;
        if(pthread_mutex_trylock(dmutex(kb))){
            // If the mutex is locked then the device is obviously in use, so keep going
            if(!strcmp(syspath, kbsyspath[index])){
                // Make sure this existing keyboard doesn't have the same syspath (this shouldn't happen)
                return 0;
            }
            continue;
        }
        if(!IS_CONNECTED(kb)){
            // Open the sysfs device
            kb->handle = open(path, O_RDWR) + 1;
            if(kb->handle <= 0){
                ckb_err("Failed to open USB device: %s\n", strerror(errno));
                kb->handle = 0;
                pthread_mutex_unlock(dmutex(kb));
                return -1;
            } else {
                // Set up device
                kb->udev = dev;
                kb->vendor = vendor;
                kb->product = product;
                strncpy(kbsyspath[index], syspath, FILENAME_MAX);
                // Mutex remains locked
                setupusb(kb);
                return 0;
            }
        }
        pthread_mutex_unlock(dmutex(kb));
    }
    ckb_err("No free devices\n");
    return -1;
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.39.3.14 static int usbclaim (usbdevice ∗ *kb*) [static]

usbclaim does claiming all EPs for the usb device gicen by kb.

Parameters

kb	THE usbdevice∗

Returns

0 on success, -1 otherwise.

Claim all endpoints for a given device (remeber the decrementing of the file descriptor) via ioctl(USBDEVFS_DISC-ONNECT) and ioctl(USBDEVFS_CLAIMINTERFACE).

Error handling is done for the ioctl(USBDEVFS_CLAIMINTERFACE) only. If this fails, now an error message is thrown and -1 is returned. Function is called in usb_linux.c only, so it is declared as static now.

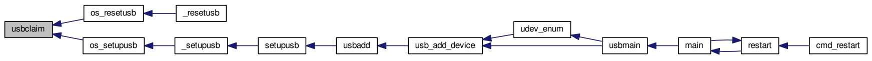
Definition at line 446 of file usb_linux.c.

References usbdevice::epcount, and usbdevice::handle.

Referenced by os_resetusb(), and os_setupusb().

```
446                             {
447     int count = kb->epcount;
448     for (int i = 0; i < count; i++) {
449         struct usbdevfs_ioctl ctl = { i, USBDEVFS_DISCONNECT, 0 };
450         ioctl(kb->handle - 1, USBDEVFS_IOCTL, &ctl);
451 
452         if (ioctl(kb->handle - 1, USBDEVFS_CLAIMINTERFACE, &i)){
453             return -1;
454         }
455     }
456     return 0;
457 }
```

Here is the caller graph for this function:



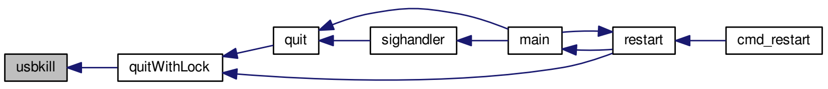
5.39.3.15 void usbkill ( )

Definition at line 803 of file usb_linux.c.

Referenced by quitWithLock().

```
803                {
804     udev_unref(udev);
805     udev = 0;
806 }
```

Here is the caller graph for this function:



5.39.3.16 int usbmain ( )

Start the USB main loop. Returns program exit code when finished.

usbmain is called by main() after setting up all other stuff.

Returns

0 normally or -1 if fatal error occurs (up to now only if no new devices are available)

First check whether the uinput module is loaded by the kernel.

Todo Why isn't missing of uinput a fatal error?

Create the udev object with udev_new() (is a function from libudev.h) terminate -1 if error

Enumerate all currently connected devices

Todo lae. here the work has to go on...

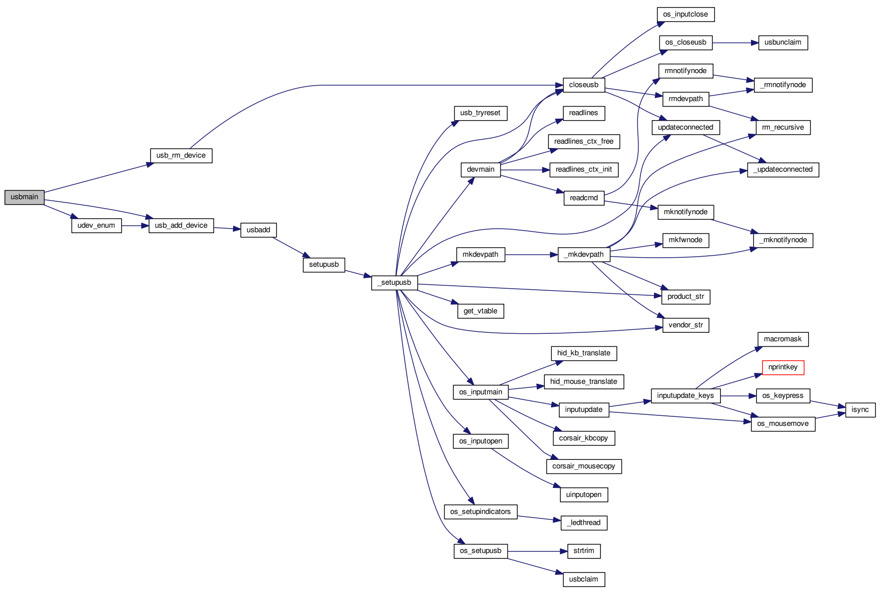
Definition at line 743 of file usb_linux.c.

References ckb_fatal, ckb_warn, udev_enum(), usb_add_device(), and usb_rm_device().

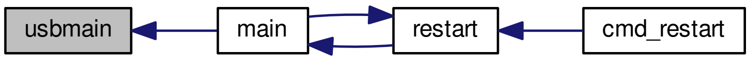
Referenced by main().

```
                {
    // Load the uinput module (if it's not loaded already)
    if(system("modprobe uinput") != 0)
        ckb_warn("Failed to load uinput module\n");

    if(!(udev = udev_new())) {
        ckb_fatal("Failed to initialize udev in usbmain(), usb_linux.c\n");
        return -1;
    }

    udev_enum();

    // Done scanning. Enter a loop to poll for device updates
    struct udev_monitor* monitor = udev_monitor_new_from_netlink(udev, "udev");
    udev_monitor_filter_add_match_subsystem_devtype(monitor, "usb", 0);
    udev_monitor_enable_receiving(monitor);
    // Get an fd for the monitor
    int fd = udev_monitor_get_fd(monitor);
    fd_set fds;
    while(udev){
        FD_ZERO(&fds);
        FD_SET(fd, &fds);
        // Block until an event is read
        if(select(fd + 1, &fds, 0, 0, 0) > 0 && FD_ISSET(fd, &fds)){
            struct udev_device* dev = udev_monitor_receive_device(monitor);
            if(!dev)
                continue;
            const char* action = udev_device_get_action(dev);
            if(!action){
                udev_device_unref(dev);
                continue;
            }
            // Add/remove device
            if(!strcmp(action, "add")){
                int res = usb_add_device(dev);
                if(res == 0)
                    continue;
                // If the device matched but the handle wasn't opened correctly, re-enumerate (this
   sometimes solves the problem)
                if(res == -1)
                    udev_enum();
            } else if(!strcmp(action, "remove"))
                usb_rm_device(dev);
            udev_device_unref(dev);
        }
    }
    udev_monitor_unref(monitor);
    return 0;
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.39.3.17 static int usbunclaim (usbdevice ∗ *kb,* int *resetting*) [static]

usbunclaim do an unclaiming of the usb device gicen by kb.

Parameters

kb	THE usbdevice∗
resetting	boolean flag: If resseting is true, the caller will perform a bus reset command after unclaiming the device.

Returns

always 0.

Unclaim all endpoints for a given device (remeber the decrementing of the file descriptor) via ioctl(USBDEVFS_DISCARDURB).

Afterwards - if ressetting is false - do a USBDEVFS_CONNECT for EP 0 and 1. If it is a non RGB device, connect EP 2 also. The comment mentions RGB keyboards only, but as I understand the code, this is valid also for RGB mice.

There is no error handling yet. Function is called in usb_linux.c only, so it is declared as static now.

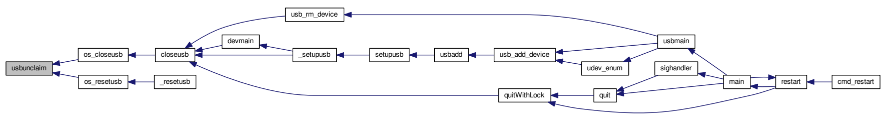
Definition at line 393 of file usb_linux.c.

References usbdevice::epcount, FEAT_RGB, usbdevice::handle, and HAS_FEATURES.

Referenced by os_closeusb(), and os_resetusb().

```
                                                {
    int handle = kb->handle - 1;
    int count = kb->epcount;
    for (int i = 0; i < count; i++) {
        ioctl(handle, USBDEVFS_RELEASEINTERFACE, &i);
    }
    // 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.
    // Reconnecting any of the others causes trouble.
    if (!resetting) {
        struct usbdevfs_ioctl ctl = { 0, USBDEVFS_CONNECT, 0 };
        ioctl(handle, USBDEVFS_IOCTL, &ctl);
        ctl.ifno = 1;
        ioctl(handle, USBDEVFS_IOCTL, &ctl);
        // Also reconnect iface #2 (HID) for non-RGB keyboards
        if(!HAS_FEATURES(kb, FEAT_RGB)){
            ctl.ifno = 2;
            ioctl(handle, USBDEVFS_IOCTL, &ctl);
        }
    }
    return 0;
}
```

Here is the caller graph for this function:



5.39.4 Variable Documentation

5.39.4.1 char kbsyspath[9][FILENAME_MAX] [static]

Definition at line 11 of file usb_linux.c.

Referenced by os_closeusb(), usb_rm_device(), and usbadd().

5.39.4.2 _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   },
    {  "1b3c" ,  0x1b3c   }
}
```

Attention

when adding new hardware this file hat to be changed too.

In this structure array *models*[] for each device the name (the device id as string in hex without leading 0x) and its usb device id as short must be entered in this array.

Definition at line 622 of file usb_linux.c.

5.39.4.3 struct udev∗ udev [static]

Definition at line 606 of file usb_linux.c.

5.39.4.4 pthread_t udevthread

Definition at line 609 of file usb_linux.c.

5.39.4.5 pthread_t usbthread

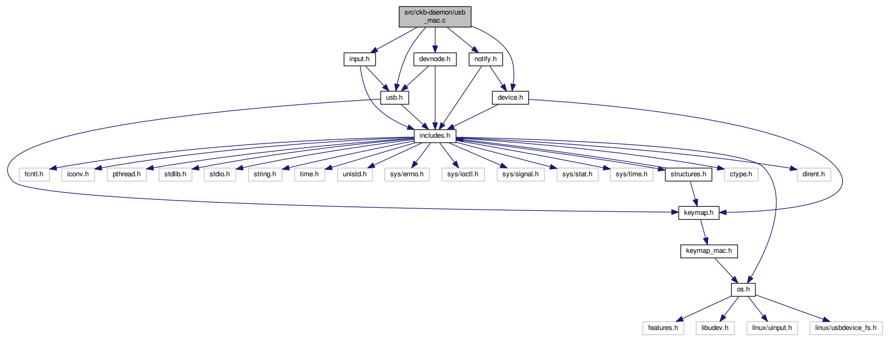
Todo These two thread vasriables seem to be unused: usbtread, udevthread

Definition at line 609 of file usb_linux.c.

5.40 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