ckb-next

v0.2.8 at branch master

Generated by Doxygen 1.8.6

Sat Jan 20 2018 16:59:28

Contents

1 README 1

2 Building ckb 9

3 cbk Improvements Roadmap 11

4 DAEMON 13

5 Todo List 21

6 Data Structure Index 23

6.1 Data Structures . . . 23

7 File Index 25

7.1 File List . . . 25

8 Data Structure Documentation 27

8.1 devcmd.__unnamed__ Struct Reference . . . 27

8.1.1 Detailed Description . . . 28

8.1.2 Field Documentation . . . 28

8.1.2.1 active . . . 28

8.1.2.2 allocprofile . . . 28

8.1.2.3 bind . . . 28

8.1.2.4 dpi . . . 28

8.1.2.5 dpisel . . . 28

8.1.2.6 erase . . . 28

8.1.2.7 eraseprofile . . . 28

8.1.2.8 freeprofile . . . 28

8.1.2.9 fwupdate . . . 28

8.1.2.10 get . . . 28

8.1.2.11 hwload . . . 28

8.1.2.12 hwsave . . . 29

8.1.2.13 iauto . . . 29

8.1.2.14 id . . . 29

8.1.2.15 idle . . . 29
8.1.2.16 inotify . . . 29
8.1.2.17 ioff . . . 29
8.1.2.18 ion . . . 29
8.1.2.19 lift . . . 29
8.1.2.20 loadprofile . . . 29
8.1.2.21 macro . . . 29
8.1.2.22 name . . . 29
8.1.2.23 notify . . . 29
8.1.2.24 pollrate . . . 29
8.1.2.25 profileid . . . 29
8.1.2.26 profilename . . . 29
8.1.2.27 rebind . . . 29
8.1.2.28 restart . . . 29
8.1.2.29 rgb . . . 29
8.1.2.30 setmodeindex . . . 29
8.1.2.31 snap . . . 29
8.1.2.32 start . . . 29
8.1.2.33 unbind . . . 29
8.1.2.34 updatedpi . . . 29
8.1.2.35 updateindicators . . . 29
8.1.2.36 updatergb . . . 29

9 File Documentation **31**

9.1 BUILD.md File Reference . . . 31
9.2 DAEMON.md File Reference . . . 31
9.3 README.md File Reference . . . 31
9.4 ROADMAP.md File Reference . . . 31
9.5 src/ckb-daemon/command.c File Reference . . . 31
9.5.1 Macro Definition Documentation . . . 32
9.5.1.1 TRY_WITH_RESET . . . 32
9.5.2 Function Documentation . . . 32
9.5.2.1 readcmd . . . 32
9.5.3 Variable Documentation . . . 36
9.5.3.1 cmd_strings . . . 36
9.6 src/ckb-daemon/command.h File Reference . . . 36
9.6.1 Data Structure Documentation . . . 38
9.6.1.1 union devcmd . . . 38
9.6.2 Macro Definition Documentation . . . 38
9.6.2.1 CMD_COUNT . . . 38

9.6.2.2 CMD_DEV_COUNT . . . 39
9.6.3 Typedef Documentation . . . 39
9.6.3.1 cmdhandler . . . 39
9.6.3.2 cmdhandler_io . . . 39
9.6.3.3 cmdhandler_mac . . . 39
9.6.3.4 devcmd . . . 39
9.6.4 Enumeration Type Documentation . . . 39
9.6.4.1 cmd . . . 39
9.6.5 Function Documentation . . . 41
9.6.5.1 readcmd . . . 41
9.7 src/ckb-daemon/device.c File Reference . . . 45
9.7.1 Function Documentation . . . 46
9.7.1.1 _start_dev . . . 46
9.7.1.2 start_dev . . . 48
9.7.2 Variable Documentation . . . 48
9.7.2.1 devlistmutex . . . 48
9.7.2.2 devmutex . . . 48
9.7.2.3 hwload_mode . . . 48
9.7.2.4 inputmutex . . . 48
9.7.2.5 keyboard . . . 48
9.7.2.6 macromutex . . . 49
9.7.2.7 macromutex2 . . . 49
9.7.2.8 macrovar . . . 49
9.8 src/ckb-daemon/device.h File Reference . . . 49
9.8.1 Macro Definition Documentation . . . 50
9.8.1.1 ACT_LIGHT . . . 50
9.8.1.2 ACT_LOCK . . . 50
9.8.1.3 ACT_M1 . . . 51
9.8.1.4 ACT_M2 . . . 51
9.8.1.5 ACT_M3 . . . 51
9.8.1.6 ACT_MR_RING . . . 51
9.8.1.7 ACT_NEXT . . . 51
9.8.1.8 ACT_NEXT_NOWRAP . . . 51
9.8.1.9 DEV_MAX . . . 51
9.8.1.10 dmutex . . . 51
9.8.1.11 imutex . . . 51
9.8.1.12 IN_CORSAIR . . . 51
9.8.1.13 IN_HID . . . 52
9.8.1.14 IS_CONNECTED . . . 52
9.8.1.15 mmutex . . . 52

9.8.1.16 mmutex2 . . . 52
9.8.1.17 mvar . . . 52
9.8.1.18 setactive . . . 52
9.8.2 Function Documentation . . . 52
9.8.2.1 cmd_active_kb . . . 52
9.8.2.2 cmd_active_mouse . . . 53
9.8.2.3 cmd_idle_kb . . . 53
9.8.2.4 cmd_idle_mouse . . . 54
9.8.2.5 cmd_pollrate . . . 54
9.8.2.6 setactive_kb . . . 55
9.8.2.7 setactive_mouse . . . 56
9.8.2.8 setmodeindex_nrgb . . . 58
9.8.2.9 start_dev . . . 58
9.8.2.10 start_kb_nrgb . . . 58
9.8.3 Variable Documentation . . . 59
9.8.3.1 devmutex . . . 59
9.8.3.2 inputmutex . . . 59
9.8.3.3 keyboard . . . 59
9.8.3.4 macromutex . . . 59
9.8.3.5 macromutex2 . . . 59
9.8.3.6 macrovar . . . 59
9.9 src/ckb-daemon/device_keyboard.c File Reference . . . 59
9.9.1 Function Documentation . . . 60
9.9.1.1 cmd_active_kb . . . 60
9.9.1.2 cmd_idle_kb . . . 61
9.9.1.3 setactive_kb . . . 61
9.9.1.4 setmodeindex_nrgb . . . 63
9.9.1.5 start_kb_nrgb . . . 63
9.10 src/ckb-daemon/device_mouse.c File Reference . . . 64
9.10.1 Function Documentation . . . 64
9.10.1.1 cmd_active_mouse . . . 64
9.10.1.2 cmd_idle_mouse . . . 65
9.10.1.3 cmd_pollrate . . . 65
9.10.1.4 setactive_mouse . . . 66
9.11 src/ckb-daemon/device_vtable.c File Reference . . . 67
9.11.1 Function Documentation . . . 68
9.11.1.1 cmd_io_none . . . 68
9.11.1.2 cmd_macro_none . . . 68
9.11.1.3 cmd_none . . . 68
9.11.1.4 int1_int_none . . . 68

9.11.1.5 int1_void_none . . . 69
9.11.1.6 loadprofile_none . . . 69
9.11.2 Variable Documentation . . . 69
9.11.2.1 vtable_keyboard . . . 69
9.11.2.2 vtable_keyboard_nonrgb . . . 69
9.11.2.3 vtable_mouse . . . 69
9.12 src/ckb-daemon/devnode.c File Reference . . . 69
9.12.1 Data Structure Documentation . . . 71
9.12.1.1 struct _readlines_ctx . . . 71
9.12.2 Macro Definition Documentation . . . 71
9.12.2.1 MAX_BUFFER . . . 71
9.12.2.2 S_GID_READ . . . 71
9.12.3 Function Documentation . . . 71
9.12.3.1 _mkdevpath . . . 72
9.12.3.2 _mknotifynode . . . 74
9.12.3.3 _rmnotifynode . . . 75
9.12.3.4 _updateconnected . . . 75
9.12.3.5 mkdevpath . . . 76
9.12.3.6 mkfwnode . . . 77
9.12.3.7 mknotifynode . . . 78
9.12.3.8 readlines . . . 78
9.12.3.9 readlines_ctx_free . . . 79
9.12.3.10 readlines_ctx_init . . . 80
9.12.3.11 rm_recursive . . . 80
9.12.3.12 rmdevpath . . . 81
9.12.3.13 rmnotifynode . . . 81
9.12.3.14 updateconnected . . . 82
9.12.4 Variable Documentation . . . 83
9.12.4.1 devpath . . . 83
9.12.4.2 gid . . . 83
9.13 src/ckb-daemon/devnode.h File Reference . . . 83
9.13.1 Macro Definition Documentation . . . 84
9.13.1.1 S_CUSTOM . . . 84
9.13.1.2 S_CUSTOM_R . . . 84
9.13.1.3 S_READ . . . 84
9.13.1.4 S_READDIR . . . 84
9.13.1.5 S_READWRITE . . . 84
9.13.2 Typedef Documentation . . . 85
9.13.2.1 readlines_ctx . . . 85
9.13.3 Function Documentation . . . 85

9.13.3.1 mkdevpath . . . 85
9.13.3.2 mkfwnode . . . 86
9.13.3.3 mknotifynode . . . 86
9.13.3.4 readlines . . . 87
9.13.3.5 readlines_ctx_free . . . 88
9.13.3.6 readlines_ctx_init . . . 88
9.13.3.7 rmdevpath . . . 89
9.13.3.8 rmnotifynode . . . 89
9.13.3.9 updateconnected . . . 90
9.13.4 Variable Documentation . . . 91
9.13.4.1 devpath . . . 91
9.13.4.2 gid . . . 91
9.14 src/ckb-daemon/dpi.c File Reference . . . 91
9.14.1 Function Documentation . . . 91
9.14.1.1 cmd_dpi . . . 91
9.14.1.2 cmd_dpisel . . . 92
9.14.1.3 cmd_lift . . . 92
9.14.1.4 cmd_snap . . . 93
9.14.1.5 loaddpi . . . 93
9.14.1.6 printdpi . . . 94
9.14.1.7 savedpi . . . 94
9.14.1.8 updatedpi . . . 95
9.15 src/ckb-daemon/dpi.h File Reference . . . 96
9.15.1 Function Documentation . . . 97
9.15.1.1 cmd_dpi . . . 97
9.15.1.2 cmd_dpisel . . . 98
9.15.1.3 cmd_lift . . . 98
9.15.1.4 cmd_snap . . . 98
9.15.1.5 loaddpi . . . 99
9.15.1.6 printdpi . . . 100
9.15.1.7 savedpi . . . 100
9.15.1.8 updatedpi . . . 101
9.16 src/ckb-daemon/extra_mac.c File Reference . . . 102
9.17 src/ckb-daemon/firmware.c File Reference . . . 103
9.17.1 Macro Definition Documentation . . . 104
9.17.1.1 FW_MAXSIZE . . . 104
9.17.1.2 FW_NOFILE . . . 104
9.17.1.3 FW_OK . . . 104
9.17.1.4 FW_USBFAIL . . . 104
9.17.1.5 FW_WRONGDEV . . . 104

9.17.2 Function Documentation . . . 104
9.17.2.1 cmd_fwupdate . . . 104
9.17.2.2 fwupdate . . . 105
9.17.2.3 getfwversion . . . 107
9.18 src/ckb-daemon/firmware.h File Reference . . . 108
9.18.1 Function Documentation . . . 108
9.18.1.1 cmd_fwupdate . . . 108
9.18.1.2 getfwversion . . . 109
9.19 src/ckb-daemon/includes.h File Reference . . . 110
9.19.1 Macro Definition Documentation . . . 112
9.19.1.1 __FILE_NOPATH__ . . . 112
9.19.1.2 ckb_err . . . 112
9.19.1.3 ckb_err_fn . . . 112
9.19.1.4 ckb_err_nofile . . . 112
9.19.1.5 ckb_fatal . . . 112
9.19.1.6 ckb_fatal_fn . . . 112
9.19.1.7 ckb_fatal_nofile . . . 112
9.19.1.8 ckb_info . . . 112
9.19.1.9 ckb_info_fn . . . 113
9.19.1.10 ckb_info_nofile . . . 113
9.19.1.11 ckb_s_err . . . 113
9.19.1.12 ckb_s_out . . . 113
9.19.1.13 ckb_warn . . . 113
9.19.1.14 ckb_warn_fn . . . 113
9.19.1.15 ckb_warn_nofile . . . 113
9.19.1.16 INDEX_OF . . . 113
9.19.1.17 timespec_eq . . . 113
9.19.1.18 timespec_ge . . . 114
9.19.1.19 timespec_gt . . . 114
9.19.1.20 timespec_le . . . 114
9.19.1.21 timespec_lt . . . 114
9.19.2 Typedef Documentation . . . 114
9.19.2.1 uchar . . . 114
9.19.2.2 ushort . . . 114
9.19.3 Function Documentation . . . 114
9.19.3.1 timespec_add . . . 114
9.20 src/ckb-daemon/input.c File Reference . . . 114
9.20.1 Macro Definition Documentation . . . 116
9.20.1.1 IS_WHEEL . . . 116
9.20.2 Function Documentation . . . 116

9.20.2.1 _cmd_macro . . . 116
9.20.2.2 cmd_bind . . . 118
9.20.2.3 cmd_macro . . . 118
9.20.2.4 cmd_rebind . . . 118
9.20.2.5 cmd_unbind . . . 119
9.20.2.6 freebind . . . 119
9.20.2.7 initbind . . . 119
9.20.2.8 inputupdate . . . 120
9.20.2.9 inputupdate_keys . . . 121
9.20.2.10 macro_pt_dequeue . . . 123
9.20.2.11 macro_pt_enqueue . . . 124
9.20.2.12 macro_pt_first . . . 125
9.20.2.13 macromask . . . 125
9.20.2.14 play_macro . . . 125
9.20.2.15 updateindicators_kb . . . 127
9.20.3 Variable Documentation . . . 128
9.20.3.1 pt_head . . . 128
9.20.3.2 pt_tail . . . 128
9.21 src/ckb-daemon/input.h File Reference . . . 128
9.21.1 Data Structure Documentation . . . 130
9.21.1.1 struct parameter . . . 130
9.21.1.2 struct ptlist . . . 131
9.21.2 Macro Definition Documentation . . . 131
9.21.2.1 IS_MOD . . . 131
9.21.3 Typedef Documentation . . . 131
9.21.3.1 parameter_t . . . 131
9.21.3.2 ptlist_t . . . 131
9.21.4 Function Documentation . . . 131
9.21.4.1 cmd_bind . . . 132
9.21.4.2 cmd_macro . . . 132
9.21.4.3 cmd_rebind . . . 132
9.21.4.4 cmd_unbind . . . 133
9.21.4.5 freebind . . . 133
9.21.4.6 initbind . . . 133
9.21.4.7 inputupdate . . . 134
9.21.4.8 os_inputclose . . . 135
9.21.4.9 os_inputopen . . . 136
9.21.4.10 os_keypress . . . 137
9.21.4.11 os_mousemove . . . 138
9.21.4.12 os_setupindicators . . . 138

9.21.4.13 updateindicators_kb . . . 139
9.22 src/ckb-daemon/input_linux.c File Reference . . . 140
9.22.1 Function Documentation . . . 140
9.22.1.1 _ledthread . . . 140
9.22.1.2 isync . . . 141
9.22.1.3 os_inputclose . . . 141
9.22.1.4 os_inputopen . . . 142
9.22.1.5 os_keypress . . . 143
9.22.1.6 os_mousemove . . . 144
9.22.1.7 os_setupindicators . . . 145
9.22.1.8 uinputopen . . . 145
9.23 src/ckb-daemon/input_mac.c File Reference . . . 146
9.24 src/ckb-daemon/input_mac_mouse.c File Reference . . . 147
9.25 src/ckb-daemon/keymap.c File Reference . . . 147
9.25.1 Macro Definition Documentation . . . 148
9.25.1.1 BUTTON_HID_COUNT . . . 148
9.25.2 Function Documentation . . . 148
9.25.2.1 corsair_kbcopy . . . 148
9.25.2.2 corsair_mousecopy . . . 148
9.25.2.3 hid_kb_translate . . . 149
9.25.2.4 hid_mouse_translate . . . 150
9.25.3 Variable Documentation . . . 151
9.25.3.1 keymap . . . 151
9.26 src/ckb-daemon/keymap.h File Reference . . . 151
9.26.1 Data Structure Documentation . . . 153
9.26.1.1 struct key . . . 153
9.26.2 Macro Definition Documentation . . . 154
9.26.2.1 BTN_WHEELDOWN . . . 154
9.26.2.2 BTN_WHEELUP . . . 154
9.26.2.3 KEY_BACKSLASH_ISO . . . 154
9.26.2.4 KEY_CORSAIR . . . 154
9.26.2.5 KEY_NONE . . . 154
9.26.2.6 KEY_UNBOUND . . . 154
9.26.2.7 LED_DPI . . . 154
9.26.2.8 LED_MOUSE . . . 154
9.26.2.9 MOUSE_BUTTON_FIRST . . . 155
9.26.2.10 MOUSE_EXTRA_FIRST . . . 155
9.26.2.11 N_BUTTONS_EXTENDED . . . 155
9.26.2.12 N_BUTTONS_HW . . . 155
9.26.2.13 N_KEY_ZONES . . . 155

9.26.2.14 N_KEYBYTES_EXTENDED . . . 155
9.26.2.15 N_KEYBYTES_HW . . . 155
9.26.2.16 N_KEYBYTES_INPUT . . . 155
9.26.2.17 N_KEYS_EXTENDED . . . 155
9.26.2.18 N_KEYS_EXTRA . . . 155
9.26.2.19 N_KEYS_HW . . . 156
9.26.2.20 N_KEYS_INPUT . . . 156
9.26.2.21 N_MOUSE_ZONES . . . 156
9.26.2.22 N_MOUSE_ZONES_EXTENDED . . . 156
9.26.2.23 SCAN_KBD . . . 156
9.26.2.24 SCAN_MOUSE . . . 156
9.26.2.25 SCAN_SILENT . . . 156
9.26.3 Function Documentation . . . 156
9.26.3.1 corsair_kbcopy . . . 156
9.26.3.2 corsair_mousecopy . . . 157
9.26.3.3 hid_kb_translate . . . 157
9.26.3.4 hid_mouse_translate . . . 159
9.26.4 Variable Documentation . . . 160
9.26.4.1 keymap . . . 160
9.27 src/ckb-daemon/keymap_mac.h File Reference . . . 160
9.28 src/ckb-daemon/led.c File Reference . . . 161
9.28.1 Function Documentation . . . 161
9.28.1.1 cmd_iauto . . . 161
9.28.1.2 cmd_inotify . . . 162
9.28.1.3 cmd_ioff . . . 162
9.28.1.4 cmd_ion . . . 163
9.28.1.5 cmd_rgb . . . 163
9.28.1.6 has_key . . . 164
9.28.1.7 iselect . . . 164
9.28.1.8 printrgb . . . 165
9.29 src/ckb-daemon/led.h File Reference . . . 166
9.29.1 Function Documentation . . . 167
9.29.1.1 cmd_iauto . . . 167
9.29.1.2 cmd_inotify . . . 168
9.29.1.3 cmd_ioff . . . 168
9.29.1.4 cmd_ion . . . 169
9.29.1.5 cmd_rgb . . . 169
9.29.1.6 loadrgb_kb . . . 170
9.29.1.7 loadrgb_mouse . . . 172
9.29.1.8 printrgb . . . 172

9.29.1.9 savergb_kb . . . 174
9.29.1.10 savergb_mouse . . . 175
9.29.1.11 updatergb_kb . . . 176
9.29.1.12 updatergb_mouse . . . 177
9.30 src/ckb-daemon/led_keyboard.c File Reference . . . 178
9.30.1 Macro Definition Documentation . . . 201
9.30.1.1 BR1 . . . 201
9.30.1.2 BR2 . . . 201
9.30.1.3 BR4 . . . 201
9.30.1.4 O0 . . . 201
9.30.1.5 O1 . . . 201
9.30.1.6 O2 . . . 202
9.30.1.7 O3 . . . 202
9.30.1.8 O4 . . . 202
9.30.1.9 O5 . . . 202
9.30.1.10 O6 . . . 202
9.30.1.11 O7 . . . 202
9.30.1.12 O8 . . . 202
9.30.2 Function Documentation . . . 202
9.30.2.1 loadrgb_kb . . . 202
9.30.2.2 makergb_512 . . . 204
9.30.2.3 makergb_full . . . 205
9.30.2.4 ordered8to3 . . . 205
9.30.2.5 quantize8to3 . . . 206
9.30.2.6 rgbcmp . . . 206
9.30.2.7 savergb_kb . . . 207
9.30.2.8 updatergb_kb . . . 208
9.30.3 Variable Documentation . . . 210
9.30.3.1 bit_reverse_table . . . 211
9.31 src/ckb-daemon/led_mouse.c File Reference . . . 212
9.31.1 Function Documentation . . . 212
9.31.1.1 isblack . . . 212
9.31.1.2 loadrgb_mouse . . . 213
9.31.1.3 rgbcmp . . . 213
9.31.1.4 savergb_mouse . . . 214
9.31.1.5 updatergb_mouse . . . 215
9.32 src/ckb-daemon/main.c File Reference . . . 216
9.32.1 Function Documentation . . . 216
9.32.1.1 localecase . . . 216
9.32.1.2 main . . . 217

9.32.1.3 quit . . . 219
9.32.1.4 quitWithLock . . . 220
9.32.1.5 restart . . . 221
9.32.1.6 sighandler . . . 222
9.32.1.7 sighandler2 . . . 223
9.32.1.8 timespec_add . . . 223
9.32.2 Variable Documentation . . . 224
9.32.2.1 features_mask . . . 224
9.32.2.2 hwload_mode . . . 224
9.32.2.3 main_ac . . . 224
9.32.2.4 main_av . . . 224
9.32.2.5 reset_stop . . . 224
9.33 src/ckb-daemon/notify.c File Reference . . . 224
9.33.1 Macro Definition Documentation . . . 225
9.33.1.1 HW_STANDARD . . . 225
9.33.1.2 HWMODE_OR_RETURN . . . 225
9.33.2 Function Documentation . . . 226
9.33.2.1 _cmd_get . . . 226
9.33.2.2 cmd_get . . . 228
9.33.2.3 cmd_notify . . . 229
9.33.2.4 cmd_restart . . . 229
9.33.2.5 nprintf . . . 230
9.33.2.6 nprintind . . . 231
9.33.2.7 nprintkey . . . 231
9.33.2.8 restart . . . 232
9.34 src/ckb-daemon/notify.h File Reference . . . 233
9.34.1 Function Documentation . . . 234
9.34.1.1 cmd_get . . . 234
9.34.1.2 cmd_notify . . . 235
9.34.1.3 cmd_restart . . . 235
9.34.1.4 nprintf . . . 235
9.34.1.5 nprintind . . . 236
9.34.1.6 nprintkey . . . 237
9.35 src/ckb-daemon/os.h File Reference . . . 238
9.35.1 Macro Definition Documentation . . . 238
9.35.1.1 _DEFAULT_SOURCE . . . 238
9.35.1.2 _GNU_SOURCE . . . 239
9.35.1.3 euid_guard_start . . . 239
9.35.1.4 euid_guard_stop . . . 239
9.35.1.5 UINPUT_VERSION . . . 239

9.36 src/ckb-daemon/profile.c File Reference . . . 239
9.36.1 Function Documentation . . . 240
9.36.1.1 _freeprofile . . . 240
9.36.1.2 allocprofile . . . 241
9.36.1.3 cmd_erase . . . 242
9.36.1.4 cmd_eraseprofile . . . 242
9.36.1.5 cmd_id . . . 243
9.36.1.6 cmd_name . . . 243
9.36.1.7 cmd_profileid . . . 244
9.36.1.8 cmd_profilename . . . 244
9.36.1.9 freemode . . . 245
9.36.1.10 freeprofile . . . 245
9.36.1.11 gethwmodename . . . 246
9.36.1.12 gethwprofilename . . . 246
9.36.1.13 getid . . . 247
9.36.1.14 getmodename . . . 248
9.36.1.15 getprofilename . . . 248
9.36.1.16 hwtonative . . . 249
9.36.1.17 initmode . . . 250
9.36.1.18 loadprofile . . . 251
9.36.1.19 nativetohw . . . 251
9.36.1.20 printname . . . 251
9.36.1.21 setid . . . 252
9.36.1.22 u16dec . . . 253
9.36.1.23 u16enc . . . 253
9.36.1.24 urldecode2 . . . 254
9.36.1.25 urlencode2 . . . 255
9.36.2 Variable Documentation . . . 255
9.36.2.1 utf16to8 . . . 256
9.36.2.2 utf8to16 . . . 256
9.37 src/ckb-daemon/profile.h File Reference . . . 256
9.37.1 Macro Definition Documentation . . . 257
9.37.1.1 hwloadprofile . . . 257
9.37.2 Function Documentation . . . 257
9.37.2.1 allocprofile . . . 257
9.37.2.2 cmd_erase . . . 258
9.37.2.3 cmd_eraseprofile . . . 258
9.37.2.4 cmd_hwload_kb . . . 259
9.37.2.5 cmd_hwload_mouse . . . 260
9.37.2.6 cmd_hwsave_kb . . . 261

9.37.2.7 cmd_hwsave_mouse . . . 262
9.37.2.8 cmd_id . . . 263
9.37.2.9 cmd_name . . . 263
9.37.2.10 cmd_profileid . . . 264
9.37.2.11 cmd_profilename . . . 264
9.37.2.12 freeprofile . . . 265
9.37.2.13 gethwmodename . . . 265
9.37.2.14 gethwprofilename . . . 266
9.37.2.15 getid . . . 267
9.37.2.16 getmodename . . . 267
9.37.2.17 getprofilename . . . 268
9.37.2.18 hwtonative . . . 269
9.37.2.19 loadprofile . . . 269
9.37.2.20 nativetohw . . . 270
9.37.2.21 setid . . . 270
9.38 src/ckb-daemon/profile_keyboard.c File Reference . . . 271
9.38.1 Function Documentation . . . 271
9.38.1.1 cmd_hwload_kb . . . 271
9.38.1.2 cmd_hwsave_kb . . . 272
9.38.1.3 hwloadmode . . . 273
9.39 src/ckb-daemon/profile_mouse.c File Reference . . . 274
9.39.1 Function Documentation . . . 274
9.39.1.1 cmd_hwload_mouse . . . 275
9.39.1.2 cmd_hwsave_mouse . . . 276
9.40 src/ckb-daemon/structures.h File Reference . . . 276
9.40.1 Data Structure Documentation . . . 278
9.40.1.1 struct usbid . . . 278
9.40.1.2 struct macroaction . . . 279
9.40.1.3 struct keymacro . . . 280
9.40.1.4 struct binding . . . 281
9.40.1.5 struct dpiset . . . 282
9.40.1.6 struct lighting . . . 283
9.40.1.7 struct usbmode . . . 284
9.40.1.8 struct usbprofile . . . 285
9.40.1.9 struct hwprofile . . . 286
9.40.1.10 struct usbinput . . . 287
9.40.1.11 struct usbdevice . . . 288
9.40.2 Macro Definition Documentation . . . 290
9.40.2.1 CLEAR_KEYBIT . . . 290
9.40.2.2 DPI_COUNT . . . 290

9.40.2.3 FEAT_ADJRATE . . . 290
9.40.2.4 FEAT_ANSI . . . 290
9.40.2.5 FEAT_BIND . . . 290
9.40.2.6 FEAT_COMMON . . . 290
9.40.2.7 FEAT_FWUPDATE . . . 290
9.40.2.8 FEAT_FWVERSION . . . 291
9.40.2.9 FEAT_HWLOAD . . . 291
9.40.2.10 FEAT_ISO . . . 291
9.40.2.11 FEAT_LMASK . . . 291
9.40.2.12 FEAT_MONOCHROME . . . 291
9.40.2.13 FEAT_MOUSEACCEL . . . 291
9.40.2.14 FEAT_NOTIFY . . . 291
9.40.2.15 FEAT_POLLRATE . . . 291
9.40.2.16 FEAT_RGB . . . 291
9.40.2.17 FEAT_STD_NRGB . . . 292
9.40.2.18 FEAT_STD_RGB . . . 292
9.40.2.19 HAS_ANY_FEATURE . . . 292
9.40.2.20 HAS_FEATURES . . . 292
9.40.2.21 HWMODE_K70 . . . 292
9.40.2.22 HWMODE_K95 . . . 292
9.40.2.23 HWMODE_MAX . . . 292
9.40.2.24 I_CAPS . . . 292
9.40.2.25 I_NUM . . . 292
9.40.2.26 I_SCROLL . . . 292
9.40.2.27 IFACE_MAX . . . 293
9.40.2.28 KB_NAME_LEN . . . 293
9.40.2.29 LIFT_MAX . . . 293
9.40.2.30 LIFT_MIN . . . 293
9.40.2.31 MACRO_MAX . . . 293
9.40.2.32 MD_NAME_LEN . . . 293
9.40.2.33 MODE_COUNT . . . 293
9.40.2.34 MSG_SIZE . . . 293
9.40.2.35 NEEDS_FW_UPDATE . . . 293
9.40.2.36 OUTFIFO_MAX . . . 294
9.40.2.37 PR_NAME_LEN . . . 294
9.40.2.38 SCROLL_ACCELERATED . . . 294
9.40.2.39 SCROLL_MAX . . . 294
9.40.2.40 SCROLL_MIN . . . 294
9.40.2.41 SERIAL_LEN . . . 294
9.40.2.42 SET_KEYBIT . . . 294

9.40.3 Variable Documentation . . . 294
9.40.3.1 vtable_keyboard . . . 294
9.40.3.2 vtable_keyboard_nonrgb . . . 294
9.40.3.3 vtable_mouse . . . 295
9.41 src/ckb-daemon/usb.c File Reference . . . 295
9.41.1 Function Documentation . . . 296
9.41.1.1 _resetusb . . . 296
9.41.1.2 _setupusb . . . 296
9.41.1.3 _usbrecv . . . 301
9.41.1.4 _usbsend . . . 302
9.41.1.5 closeusb . . . 303
9.41.1.6 devmain . . . 305
9.41.1.7 get_vtable . . . 307
9.41.1.8 product_str . . . 308
9.41.1.9 revertusb . . . 309
9.41.1.10 setupusb . . . 310
9.41.1.11 usb_tryreset . . . 311
9.41.1.12 vendor_str . . . 312
9.41.2 Variable Documentation . . . 312
9.41.2.1 features_mask . . . 312
9.41.2.2 hwload_mode . . . 312
9.41.2.3 reset_stop . . . 313
9.41.2.4 usbmutex . . . 313
9.42 src/ckb-daemon/usb.h File Reference . . . 313
9.42.1 Detailed Description . . . 317
9.42.2 Macro Definition Documentation . . . 317
9.42.2.1 DELAY_LONG . . . 317
9.42.2.2 DELAY_MEDIUM . . . 317
9.42.2.3 DELAY_SHORT . . . 318
9.42.2.4 IS_FULLRANGE . . . 318
9.42.2.5 IS_GLAIVE . . . 318
9.42.2.6 IS_HARPOON . . . 318
9.42.2.7 IS_K63 . . . 318
9.42.2.8 IS_K65 . . . 318
9.42.2.9 IS_K68 . . . 318
9.42.2.10 IS_K70 . . . 318
9.42.2.11 IS_K95 . . . 318
9.42.2.12 IS_M65 . . . 319
9.42.2.13 IS_MONOCHROME . . . 319
9.42.2.14 IS_MONOCHROME_DEV . . . 319

9.42.2.15 IS_MOUSE . . . 319
9.42.2.16 IS_MOUSE_DEV . . . 319
9.42.2.17 IS_PLATINUM . . . 319
9.42.2.18 IS_RGB . . . 319
9.42.2.19 IS_RGB_DEV . . . 319
9.42.2.20 IS_SABRE . . . 319
9.42.2.21 IS_SCIMITAR . . . 320
9.42.2.22 IS_STRAFE . . . 320
9.42.2.23 IS_V2_OVERRIDE . . . 320
9.42.2.24 NK95_HWOFF . . . 320
9.42.2.25 NK95_HWON . . . 320
9.42.2.26 NK95_M1 . . . 320
9.42.2.27 NK95_M2 . . . 320
9.42.2.28 NK95_M3 . . . 320
9.42.2.29 nk95cmd . . . 321
9.42.2.30 P_GLAIVE . . . 321
9.42.2.31 P_GLAIVE_STR . . . 321
9.42.2.32 P_HARPOON . . . 321
9.42.2.33 P_HARPOON_STR . . . 321
9.42.2.34 P_K63_NRGB . . . 321
9.42.2.35 P_K63_NRGB_STR . . . 321
9.42.2.36 P_K65 . . . 321
9.42.2.37 P_K65_LUX . . . 321
9.42.2.38 P_K65_LUX_STR . . . 321
9.42.2.39 P_K65_NRGB . . . 322
9.42.2.40 P_K65_NRGB_STR . . . 322
9.42.2.41 P_K65_RFIRE . . . 322
9.42.2.42 P_K65_RFIRE_STR . . . 322
9.42.2.43 P_K65_STR . . . 322
9.42.2.44 P_K68 . . . 322
9.42.2.45 P_K68_STR . . . 322
9.42.2.46 P_K70 . . . 322
9.42.2.47 P_K70_LUX . . . 322
9.42.2.48 P_K70_LUX_NRGB . . . 322
9.42.2.49 P_K70_LUX_NRGB_STR . . . 323
9.42.2.50 P_K70_LUX_STR . . . 323
9.42.2.51 P_K70_NRGB . . . 323
9.42.2.52 P_K70_NRGB_STR . . . 323
9.42.2.53 P_K70_RFIRE . . . 323
9.42.2.54 P_K70_RFIRE_NRGB . . . 323

9.42.2.55 P_K70_RFIRE_NRGB_STR . . . 323
9.42.2.56 P_K70_RFIRE_STR . . . 323
9.42.2.57 P_K70_STR . . . 323
9.42.2.58 P_K95 . . . 323
9.42.2.59 P_K95_NRGB . . . 323
9.42.2.60 P_K95_NRGB_STR . . . 324
9.42.2.61 P_K95_PLATINUM . . . 324
9.42.2.62 P_K95_PLATINUM_STR . . . 324
9.42.2.63 P_K95_STR . . . 324
9.42.2.64 P_M65 . . . 324
9.42.2.65 P_M65_PRO . . . 324
9.42.2.66 P_M65_PRO_STR . . . 324
9.42.2.67 P_M65_STR . . . 324
9.42.2.68 P_SABRE_L . . . 324
9.42.2.69 P_SABRE_L_STR . . . 324
9.42.2.70 P_SABRE_N . . . 324
9.42.2.71 P_SABRE_N_STR . . . 325
9.42.2.72 P_SABRE_O . . . 325
9.42.2.73 P_SABRE_O2 . . . 325
9.42.2.74 P_SABRE_O2_STR . . . 325
9.42.2.75 P_SABRE_O_STR . . . 325
9.42.2.76 P_SCIMITAR . . . 325
9.42.2.77 P_SCIMITAR_PRO . . . 325
9.42.2.78 P_SCIMITAR_PRO_STR . . . 325
9.42.2.79 P_SCIMITAR_STR . . . 325
9.42.2.80 P_STRAFE . . . 325
9.42.2.81 P_STRAFE_NRGB . . . 326
9.42.2.82 P_STRAFE_NRGB_2 . . . 326
9.42.2.83 P_STRAFE_NRGB_2_STR . . . 326
9.42.2.84 P_STRAFE_NRGB_STR . . . 326
9.42.2.85 P_STRAFE_STR . . . 326
9.42.2.86 resetusb . . . 326
9.42.2.87 USB_DELAY_DEFAULT . . . 326
9.42.2.88 usbrecv . . . 326
9.42.2.89 usbsend . . . 326
9.42.2.90 V_CORSAIR . . . 327
9.42.2.91 V_CORSAIR_STR . . . 327
9.42.3 Function Documentation . . . 327
9.42.3.1 _nk95cmd . . . 327
9.42.3.2 _resetusb . . . 329

9.42.3.3 _usbrecv . . . 329
9.42.3.4 _usbsend . . . 331
9.42.3.5 closeusb . . . 333
9.42.3.6 os_closeusb . . . 334
9.42.3.7 os_inputmain . . . 335
9.42.3.8 os_resetusb . . . 339
9.42.3.9 os_sendindicators . . . 340
9.42.3.10 os_setupusb . . . 341
9.42.3.11 os_usbrecv . . . 343
9.42.3.12 os_usbsend . . . 345
9.42.3.13 product_str . . . 347
9.42.3.14 revertusb . . . 348
9.42.3.15 setupusb . . . 349
9.42.3.16 usb_tryreset . . . 350
9.42.3.17 usbkill . . . 351
9.42.3.18 usbmain . . . 352
9.42.3.19 vendor_str . . . 353
9.43 src/ckb-daemon/usb_linux.c File Reference . . . 354
9.43.1 Data Structure Documentation . . . 356
9.43.1.1 struct _model . . . 356
9.43.2 Macro Definition Documentation . . . 356
9.43.2.1 DEBUG . . . 356
9.43.2.2 N_MODELS . . . 356
9.43.2.3 TEST_RESET . . . 356
9.43.3 Function Documentation . . . 357
9.43.3.1 _nk95cmd . . . 357
9.43.3.2 os_closeusb . . . 358
9.43.3.3 os_inputmain . . . 358
9.43.3.4 os_resetusb . . . 362
9.43.3.5 os_sendindicators . . . 363
9.43.3.6 os_setupusb . . . 364
9.43.3.7 os_usbrecv . . . 366
9.43.3.8 os_usbsend . . . 367
9.43.3.9 strtrim . . . 369
9.43.3.10 udev_enum . . . 369
9.43.3.11 usb_add_device . . . 371
9.43.3.12 usb_rm_device . . . 373
9.43.3.13 usbadd . . . 374
9.43.3.14 usbclaim . . . 375
9.43.3.15 usbkill . . . 376

9.43.3.16 usbmain . . . 376
9.43.3.17 usbunclaim . . . 378
9.43.4 Variable Documentation . . . 379
9.43.4.1 kbsyspath . . . 379
9.43.4.2 models . . . 379
9.43.4.3 udev . . . 379
9.43.4.4 udevthread . . . 379
9.43.4.5 usbthread . . . 380
9.44 src/ckb-daemon/usb_mac.c File Reference . . . 380

Index **381**

Chapter 1

README

ckb-next: RGB Driver for Linux and macOS

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

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

- Device Support
  - Keyboards
  - Mice
- Linux Installation
  - Pre-made packages
  - Preparation
  - Installing
  - Upgrading
  - Uninstalling
- OS X/macOS Installation
  - Binary download
  - Building from source
  - Upgrading (binary)
  - Upgrading (source)
  - Uninstalling
- Usage
  - Major features
  - Roadmap
- Troubleshooting
  - Linux

    - OS X/macOS
    - General
    - Reporting issues
- Known issues
- Contributing
- Contact us
- What happened to the original ckb

See also:

- https://github.com/mattanger/ckb-next/blob/master/DAEMON.md "Manual for the driver daemon"

Device Support

Keyboards

- K63
- K65:
    - RGB
    - non-RGB
    - LUX RGB
    - RGB RAPIDFIRE
- K68
- K70:
    - RGB
    - non-RGB
    - LUX RGB
    - LUX non-RGB
    - RGB RAPIDFIRE
    - non-RGB RAPIDFIRE
- K95:
    - RGB
    - non-RGB∗
    - Platinum∗∗
- Strafe:
    - RGB
    - non-RGB
- = hardware playback not supported. Settings will be saved to software only.

∗∗ = partial support, static hardware playback only.

Mice

- M65:
  - non-RGB
  - PRO RGB
- Sabre:
  - Optical RGB
  - Laser RGB
- Scimitar:
  - RGB
  - PRO RGB
- Harpoon
- Glaive

Linux Installation

Pre-made packages

- Fedora 24/25, CentOS/RHEL 7 (maintained by):
  - `'johanh/ckb'` - based on `master` branch
- Arch Linux (maintained by ,):
  - `'aur/ckb-next'` - based on GitHub releases
  - `'aur/ckb-next-git'` - based on `master` branch
  - `'aur/ckb-next-latest-git'` - based on `newdev` branch
  - `'aur/ckb-next-newfw-git'` - based on `newfw` branch

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

Preparation

ckb-next requires Qt5 (Qt 5.9 is recommended), libudev, zlib, gcc, g++, and glibc.

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

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

Installing

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

Upgrading

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

Uninstalling

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

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

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

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

OS X/macOS Installation

Binary download

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

Building from source

Install the latest version of Xcode from the App Store. While it's downloading, open the Terminal and execute `xcode-select --install` to install Command Line Tools. Then open Xcode, accept the license agreement and wait for it to install any additional components (if necessary). When you see the "Welcome to Xcode" screen, from the top bar choose `Xcode -> Preferences -> Locations -> Command Line Tools` and select an SDK version. Afterwards install `Homebrew` and execute `brew install qt5` in the Terminal.

> **Note**: If you decide to use the official Qt5 package from Qt website instead, you will have to edit the installation script and provide installation paths manually due to a qmake bug.

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

Upgrading (binary)

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

Upgrading (source)

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

Uninstalling

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

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

Usage

The user interface is still a work in progress.

Major features

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

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

Roadmap

- **v0.3 release:**
- Ability to store profiles separately from devices, import/export them
- More functions for the Win Lock key
- Key macros
- **v0.4 release:**
- Ability to import CUE profiles

- Ability to tie profiles to which application has focus
- **v0.5 release:**
- Key combos
- Timers?
- **v1.0 release:**
- OSD? (Not sure if this can actually be done)
- Extra settings?
- ????

Troubleshooting

Linux

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

- K65 RGB: `usbhid.quirks=0x1B1C:0x1B17:0x20000408`
- K65 LUX: `usbhid.quirks=0x1B1C:0x1B37:0x20000408`
- K68: `usbhid.quirks=0x1B1C:0x1B3F:0x20000408`
- K70: `usbhid.quirks=0x1B1C:0x1B09:0x20000408`
- K70 LUX: `usbhid.quirks=0x1B1C:0x1B36:0x20000408`
- K70 RGB: `usbhid.quirks=0x1B1C:0x1B13:0x20000408`
- K95: `usbhid.quirks=0x1B1C:0x1B08:0x20000408`
- K95 RGB: `usbhid.quirks=0x1B1C:0x1B11:0x20000408`
- Strafe: `usbhid.quirks=0x1B1C:0x1B15:0x20000408`
- Strafe RGB: `usbhid.quirks=0x1B1C:0x1B20:0x20000408`
- M65 RGB: `usbhid.quirks=0x1B1C:0x1B12:0x20000408`
- Sabre RGB Optical: `usbhid.quirks=0x1B1C:0x1B14:0x20000408`
- Sabre RGB Laser: `usbhid.quirks=0x1B1C:0x1B19:0x20000408`
- Scimitar RGB: `usbhid.quirks=0x1B1C:0x1B1E:0x20000408`

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

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

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

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

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

Fedora 26 Color Changer Freeze Fix

If you're running Fedora 26, a working solution for the color changer freezing issue is to install qt5ct `dnf install qt5ct` then modify your /etc/environment file to contain the line `QT_QPA_PLATFORMTHEME=qt5ct`

OS X/macOS

- ∗∗"ckb.pkg" can't be opened because it is from an unidentified developer∗∗ Right-click (control-click) on ckb.pkg and select Open. This new dialog box will give you the option to open anyway, without changing your system preferences.
- **Modifier keys (Shift, Ctrl, etc.) are not rebound correctly** ckb does not recognize modifier keys rebound from System Preferences. You can rebind them again within the application.
- ∗∗∼ key prints `§±`∗∗ Check your keyboard layout on ckb's Settings screen. Choose the layout that matches your physical keyboard.
- **Compile problems** Can usually be resolved by rebooting your computer and/or reinstalling Qt. Make sure that Xcode works on its own. If a compile fails, delete the `ckb-master` directory as well as any automatically generated `build-ckb` folders and try again from a new download.
- **Scroll wheel does not scroll** As of #c3474d2 it's now possible to **disable scroll acceleration** from the GUI. You can access it under "OSX tweaks" in the "More settings" screen. Once disabled, the scroll wheel should behave consistently.

General

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

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

Common issues:

- **Problem:** ckb says "No devices connected" or "Driver inactive"
- **Solution:** Try rebooting the computer and/or reinstalling ckb. Try removing the keyboard and plugging it back in. If the error doesn't go away, try the following:
- **Problem:** Keyboard doesn't work in BIOS, doesn't work at boot
- **Solution:** Some BIOSes have trouble communicating with the keyboard. They may prevent the keyboard from working correctly in the operating system as well. First, try booting the OS *without* the keyboard attached, and plug the keyboard in after logging in. If the keyboard works after the computer is running but does not work at boot, you may need to use the keyboard's BIOS mode option.
- BIOS mode can be activated using the poll rate switch at the back of the keyboard. Slide it all the way to the position marked "BIOS". You should see the scroll lock light blinking to indicate that it is on. (Note: Unfortunately, this has its own problems - see Known Issues. You may need to activate BIOS mode when booting the computer and deactivate it after logging in).
- **Problem:** Keyboard isn't detected when plugged in, even if driver is already running
- **Solution:** Try moving to a different USB port. Be sure to follow `Corsair's USB connection requirements`. Note that the keyboard does not work with some USB3 controllers - if you have problems with USB3 ports, try USB2 instead. If you have any USB hubs on hand, try those as well. You may also have success sliding the poll switch back and forth a few times.

Reporting issues

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

Known issues

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

Contributing

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

Contact us

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

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

What happened to the original ckb

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

Chapter 2

Building ckb

Linux

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

Running as a service:

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

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

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

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

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

Running manually:

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

OSX

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

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

Running as a service:

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

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

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

Running manually:

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

Chapter 3

cbk Improvements Roadmap

Short term plan

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

Chapter 4

DAEMON

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

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

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

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

Other `ckb*` devices contain the following:

- `cmd`: Keyboard controller.
- `notify0`: Keyboard- or mouse notifications.
- `notify1`: Keyboard- or mouse notifications, used for macro recording.
- `features`: Device features.
- `fwversion`: Device firmware version (not present on all devices).
- `model`: Device description/model.
- `pollrate`: Poll rate in milliseconds (not present on all devices).
- `productid`: Contains the USB productID of the hardware
- `serial`: Device serial number. `model` and `serial` will match the info found in `ckb0/connected`

Commands

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

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

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

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

Features

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

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

Keyboard layout

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

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

Poll rate

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

Profiles and modes

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

- profilename <name> sets the profile's name. The name must be written without spaces; to add a space, use %20.

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

Examples:

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

LED commands

The backlighting is controlled by the `rgb` commands.

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

Examples:

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

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

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

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

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

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

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

Indicators

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

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

Binding keys

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

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

Examples:

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

Key macros

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

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

Examples:

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

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

Macro playback delay

There are two types of playback delay that can be set with macros; global and local. Setting a *global delay* value introduces a time delay between events during macro execution or playback. *Local delay* allows setting the delay after an individual event, overriding the global delay value for that event. Thus global delay can be used to set the overall playback speed of macros and local delays can be used to tune individual events within a macro.

All delay values are specified in microseconds (us) and are positive values from `0` to `UINT_MAX - 1`. This means delays range from 0 to just over 1 hour (4,294,967,294us, 4,294 seconds, 71 minutes, or 1.19 hours). A value of zero (0) represents no delay between actions.

Global macro delay (default delay)

Global delay allows macro playback speed to be changed. It sets the time between (actually after) each recorded macro event. If global delay is set to 1 microsecond then a 1 ms delay will follow each individual macro event when the macro is triggered.

The *global delay* is set with the ckb-daemon's existing (in testing branch) `delay` command followed by an unsigned integer representing the number of microseconds to wait after each macro action and before the next.

Global delay can also be set to `on` which maintains backwards compatibility with the current development of `ckb-daemon` for long macro playback. That is, setting the global delay to `on` introduces a 30us and a 100us delay based on the macro's length during playback.

NOTE: This setting also introduces a delay after the last macro action. This functionality exists in the current testing branch and was left as-is. It is still to be determined if this is a bug or a feature.

Examples:

- `delay 1000` sets a 1,000us delay between action playback.
- `delay on` sets long macro delay; 30us for actions between 20 and 200, 100us for actions > 200.
- `delay off` sets no delay (same as 0).
- `delay 0` sets no delay (same as off).
- `delay spearmint-potato` is invalid input, sets no delay (same as off).

Local macro delay (keystroke delay)

Local Delay allows each macro action to have a post-action delay associated with it. This allows a macro to vary it's playback speed for each event. If no local delay is specified for a macro action, then the global `delay` (above) is used. All delay values are in microsecods (us) as with the global delay setting.

Examples:

- `macro g5:+d,-d,+e=5000,-e,+l,-l=10000,+a,-a,+y,-y=1000000,+enter,-enter` define a macro for `g5` with a 5,000us delay between the `e` down and `e` up actions. A 1,000us delay between `l` up and `a` down, a delay of one second (1,000,000us) after `y` up and before `enter`, and the global delay for all other actions.
- `macro g5:+d,-d=0` use default delay between `d` down and `d` up and no delay (0us) after `d` up. This removes the noted feature/bug (above) where the last action has a trailing delay associated with it.

DPI and mouse settings

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

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

- `dpi <stage>:off` disables a DPI stage.
- `dpisel <stage>` sets the current stage selection.

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

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

Additional mouse settings:

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

Notifications

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

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

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

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

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

Examples:

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

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

Indicator notifications

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

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

Getting parameters

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

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

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

Firmware updates

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

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

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

Restart

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

There are two ways to restart the daemon:

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

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

Security

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

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

Chapter 5

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 inputupdate_keys (usbdevice ∗kb)

If we want to get all keys typed while a macro is played, add the code for it here.

Global macro_pt_enqueue ()

find a better exit strategy if no more mem available.

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 6

Data Structure Index

6.1 Data Structures

Here are the data structures with brief descriptions:

devcmd.__unnamed__ 27

Chapter 7

File Index

7.1 File List

Here is a list of all files with brief descriptions:

src/ckb-daemon/command.c . . . 31
src/ckb-daemon/command.h . . . 36
src/ckb-daemon/device.c . . . 45
src/ckb-daemon/device.h . . . 49
src/ckb-daemon/device_keyboard.c . . . 59
src/ckb-daemon/device_mouse.c . . . 64
src/ckb-daemon/device_vtable.c . . . 67
src/ckb-daemon/devnode.c . . . 69
src/ckb-daemon/devnode.h . . . 83
src/ckb-daemon/dpi.c . . . 91
src/ckb-daemon/dpi.h . . . 96
src/ckb-daemon/extra_mac.c . . . 102
src/ckb-daemon/firmware.c . . . 103
src/ckb-daemon/firmware.h . . . 108
src/ckb-daemon/includes.h . . . 110
src/ckb-daemon/input.c . . . 114
src/ckb-daemon/input.h . . . 128
src/ckb-daemon/input_linux.c . . . 140
src/ckb-daemon/input_mac.c . . . 146
src/ckb-daemon/input_mac_mouse.c . . . 147
src/ckb-daemon/keymap.c . . . 147
src/ckb-daemon/keymap.h . . . 151
src/ckb-daemon/keymap_mac.h . . . 160
src/ckb-daemon/led.c . . . 161
src/ckb-daemon/led.h . . . 166
src/ckb-daemon/led_keyboard.c . . . 178
src/ckb-daemon/led_mouse.c . . . 212
src/ckb-daemon/main.c . . . 216
src/ckb-daemon/notify.c . . . 224
src/ckb-daemon/notify.h . . . 233
src/ckb-daemon/os.h . . . 238
src/ckb-daemon/profile.c . . . 239
src/ckb-daemon/profile.h . . . 256
src/ckb-daemon/profile_keyboard.c . . . 271
src/ckb-daemon/profile_mouse.c . . . 274
src/ckb-daemon/structures.h . . . 276
src/ckb-daemon/usb.c . . . 295

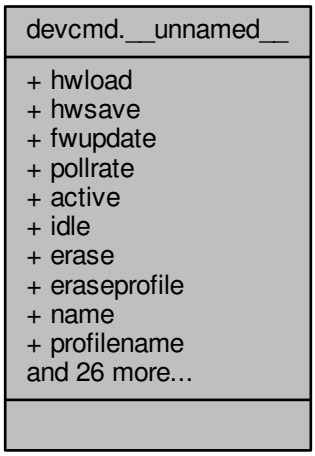
src/ckb-daemon/usb.h
Definitions for using USB interface 313
src/ckb-daemon/usb_linux.c 354
src/ckb-daemon/usb_mac.c 380

Chapter 8

Data Structure Documentation

8.1 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)

8.1.1 Detailed Description

Definition at line 78 of file command.h.

8.1.2 Field Documentation

8.1.2.1

8.1.2.2

8.1.2.3

8.1.2.4

8.1.2.5

8.1.2.6

8.1.2.7

8.1.2.8

8.1.2.9

8.1.2.10

8.1.2.11

8.1.2.12

8.1.2.13

8.1.2.14

8.1.2.15

8.1.2.16

8.1.2.17

8.1.2.18

8.1.2.19

8.1.2.20

8.1.2.21

8.1.2.22

8.1.2.23

8.1.2.24

8.1.2.25

8.1.2.26

8.1.2.27

8.1.2.28

8.1.2.29

8.1.2.30

8.1.2.31

8.1.2.32

8.1.2.33

8.1.2.34

8.1.2.35

8.1.2.36

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

Chapter 9

File Documentation

9.1 BUILD.md File Reference

9.2 DAEMON.md File Reference

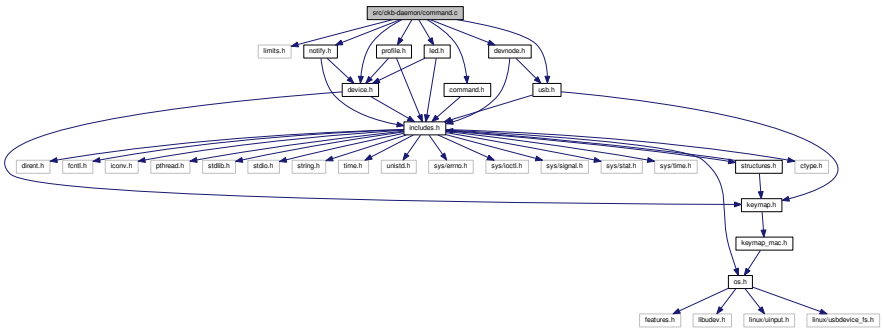
9.3 README.md File Reference

9.4 ROADMAP.md File Reference

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

```
#include <limits.h>
#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]

9.5.1 Macro Definition Documentation

9.5.1.1 #define TRY_WITH_RESET(*action*)

Value:

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

Definition at line 59 of file command.c.

Referenced by readcmd().

9.5.2 Function Documentation

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

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

Definition at line 68 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().

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

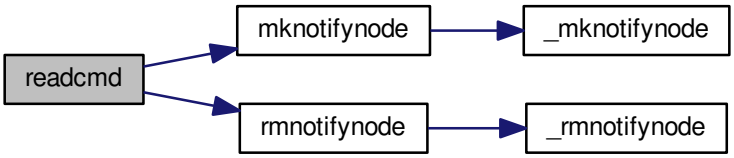
```
88          }
89          // Check for a command word
90          for(int i = 0; i < CMD_COUNT - 1; i++){
91              if(!strcmp(word, cmd_strings[i])){
92                  command = i + CMD_FIRST;
93  #ifndef OS_MAC
94                  // Layout and mouse acceleration aren't used on Linux; ignore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

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

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

```
335                         vt->do_cmd[command](kb, mode, notifynumber, i, right);
336                         break;
337                     }
338                 }
339             }
340             if(word[position += field] == ',')
341                 position++;
342         }
343     }
344
345     // Finish up
346     if(!NEEDS_FW_UPDATE(kb)){
347         TRY_WITH_RESET(vt->updatergb(kb, 0));
348         TRY_WITH_RESET(vt->updatedpi(kb, 0));
349     }
350     free(word);
351     return 0;
352 }
```

Here is the call graph for this function:



Here is the caller graph for this function:

9.5.3 Variable Documentation

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

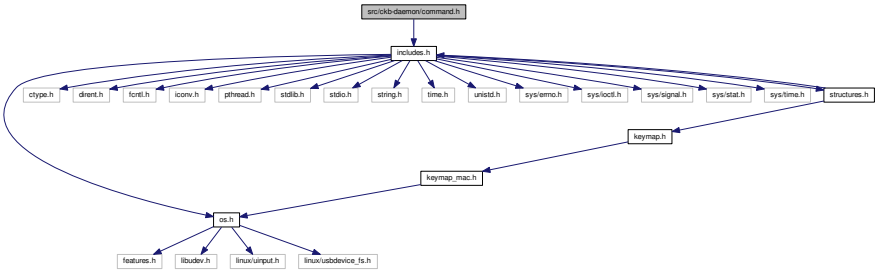
Definition at line 10 of file command.c.

Referenced by readcmd().

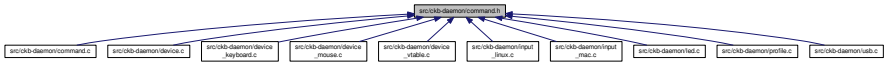
9.6 src/ckb-daemon/command.h File Reference

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

Include dependency graph for command.h:



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



Data Structures

- union devcmd
- struct devcmd.__unnamed__

Macros

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

Typedefs

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

Enumerations

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

Functions

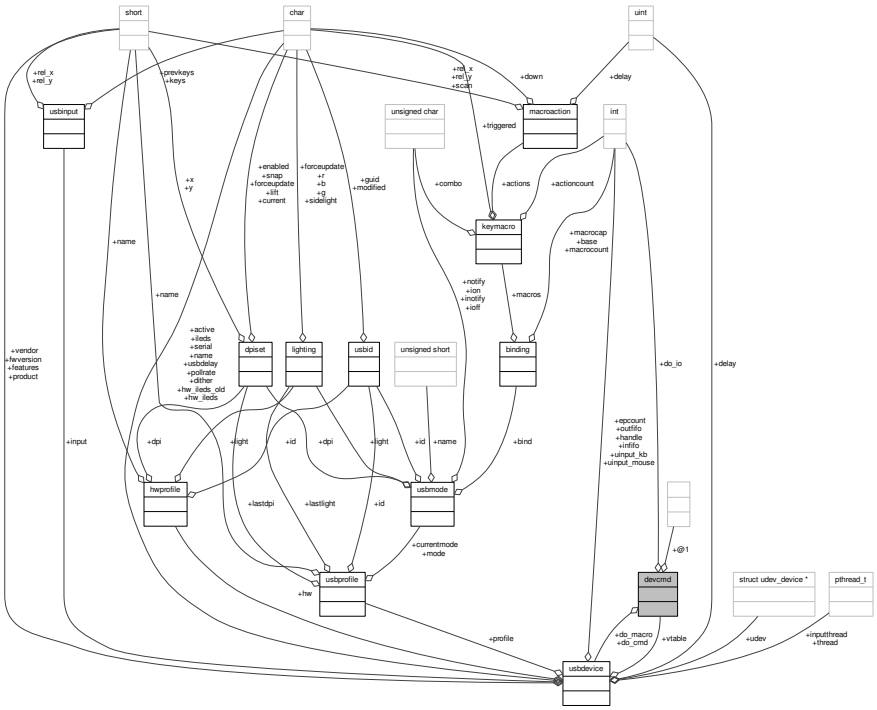
- int readcmd (usbdevice *kb, const char *line)

9.6.1 Data Structure Documentation

9.6.1.1 union devcmd

Definition at line 73 of file command.h.

Collaboration diagram for devcmd:



Data Fields

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

9.6.2 Macro Definition Documentation

9.6.2.1 #define CMD_COUNT (CMD_LAST - CMD_FIRST + 2)

Definition at line 65 of file command.h.

Referenced by readcmd().

9.6.2.2 #define CMD_DEV_COUNT (CMD_LAST - CMD_VT_FIRST + 1)

Definition at line 66 of file command.h.

9.6.3 Typedef Documentation

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

Definition at line 70 of file command.h.

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

Definition at line 71 of file command.h.

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

Definition at line 72 of file command.h.

9.6.3.4 typedef union devcmd devcmd

9.6.4 Enumeration Type Documentation

9.6.4.1 enum cmd

Enumerator

NONE

DELAY

CMD_FIRST

MODE

SWITCH

LAYOUT

ACCEL

SCROLLSPEED

NOTIFYON

NOTIFYOFF

FPS

DITHER

HWLOAD

CMD_VT_FIRST

HWSAVE

FWUPDATE

POLLRATE

ACTIVE

IDLE

ERASE

ERASEPROFILE

NAME

PROFILENAME

ID

PROFILEID

RGB

IOFF

ION

IAUTO

BIND

UNBIND

REBIND

MACRO

DPI

DPISEL

LIFT

SNAP

NOTIFY

INOTIFY

GET

RESTART

CMD_LAST

Definition at line 7 of file command.h.

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

```
47     UNBIND,
48     REBIND,
49     MACRO,
50
51     // DPI control
52     DPI,
53     DPISEL,
54     LIFT,
55     SNAP,
56
57     // Notifications and output
58     NOTIFY,
59     INOTIFY,
60     GET,
61     RESTART,
62
63     CMD_LAST = RESTART
64 } cmd;
```

9.6.5 Function Documentation

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

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

Definition at line 68 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().

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

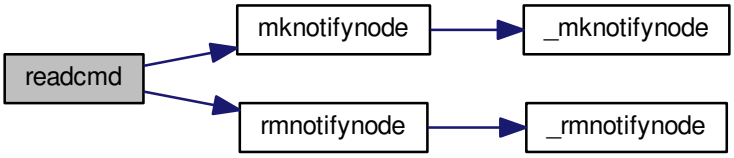
```
104                     goto next_loop;
105                 break;
106             }
107         }
108 
109         // Set current notification node when given @number
110         int newnotify;
111         if(sscanf(word, "@%u", &newnotify) == 1 && newnotify < OUTFIFO_MAX){
112             notifynumber = newnotify;
113             continue;
114         }
115 
116         // Reject unrecognized commands. Reject bind or notify related commands if the keyboard doesn't
     have the feature enabled.
117         if(command == NONE
118                 || ((!HAS_FEATURES(kb, FEAT_BIND) && (command ==
    BIND || command == UNBIND || command == REBIND || command ==
    MACRO || command == DELAY))
119                         || (!HAS_FEATURES(kb, FEAT_NOTIFY) && command ==
    NOTIFY))){
120             next_loop:
121             continue;
122         }
123         // Reject anything not related to fwupdate if device has a bricked FW
124         if(NEEDS_FW_UPDATE(kb) && command != FWUPDATE && command !=
    NOTIFYON && command != NOTIFYOFF)
125             continue;
126 
127         // Specially handled commands - these are available even when keyboard is IDLE
128         switch(command){
129         case NOTIFYON: {
130             // Notification node on
131             int notify;
132             if(sscanf(word, "%u", &notify) == 1)
133                 mknotifynode(kb, notify);
134             continue;
135         } case NOTIFYOFF: {
136             // Notification node off
137             int notify;
138             if(sscanf(word, "%u", &notify) == 1 && notify != 0) // notify0 can't be removed
139                 rmnotifynode(kb, notify);
140             continue;
141         } case GET:
142             // Output data to notification node
143             vt->get(kb, mode, notifynumber, 0, word);
144             continue;
145         case LAYOUT:
146             // OSX: switch ANSI/ISO keyboard layout
147             if(!strcmp(word, "ansi"))
148                 kb->features = (kb->features & ~FEAT_LMASK) |
    FEAT_ANSI;
149             else if(!strcmp(word, "iso"))
150                 kb->features = (kb->features & ~FEAT_LMASK) |
    FEAT_ISO;
151             continue;
152 #ifdef OS_MAC
153         case ACCEL:
154             // OSX mouse acceleration on/off
155             if(!strcmp(word, "on"))
156                 kb->features |= FEAT_MOUSEACCEL;
157             else if(!strcmp(word, "off"))
158                 kb->features &= ~FEAT_MOUSEACCEL;
159             continue;
160         case SCROLLSPEED:{
161             int newscroll;
162             if(sscanf(word, "%d", &newscroll) != 1)
163                 break;
164             if(newscroll < SCROLL_MIN)
165                 newscroll = SCROLL_ACCELERATED;
166             if(newscroll > SCROLL_MAX)
167                 newscroll = SCROLL_MAX;
168             kb->scroll_rate = newscroll;
169             continue;
170         }
171 #endif
172         case MODE: {
173             // Select a mode number (1 - 6)
174             int newmode;
175             if(sscanf(word, "%u", &newmode) == 1 && newmode > 0 && newmode <=
    MODE_COUNT)
176                 mode = profile->mode + newmode - 1;
177             continue;
178         }
179         case FPS: {
180             // USB command delay (2 - 10ms)
181             uint framerate;
182             if(sscanf(word, "%u", &framerate) == 1 && framerate > 0){
```

```
183                 // Not all devices require the same number of messages per frame; select delay
    appropriately
184                 uint per_frame = IS_MOUSE_DEV(kb) ? 2 : IS_FULLRANGE(kb) ? 14 : 5;
185                 uint delay = 1000 / framerate / per_frame;
186                 if(delay < 2)
187                     delay = 2;
188                 else if(delay > 10)
189                     delay = 10;
190                 kb->usbdelay = delay;
191             }
192             continue;
193         }
194         case DITHER: {
195             // 0: No dither, 1: Ordered dither.
196             uint dither;
197             if(sscanf(word, "%u", &dither) == 1 && dither <= 1){
198                 kb->dither = dither;
199                 profile->currentmode->light.forceupdate = 1;
200                 mode->light.forceupdate = 1;
201             }
202             continue;
203         }
204         case DELAY: {
205             long int delay;
206             if(sscanf(word, "%ld", &delay) == 1 && 0 <= delay && delay < UINT_MAX) {
207                 // Add delay of `newdelay` microseconds to macro playback
208                 kb->delay = (unsigned int)delay;
209             } else if(strcmp(word, "on") == 0) {
210                 // allow previous syntax, `delay on` means use old `long macro delay`
211                 kb->delay = UINT_MAX;
212             } else {
213                 // bad parameter to handle false commands like "delay off"
214                 kb->delay = 0; // No delay.
215             }
216             continue;
217         }
218         case RESTART: {
219             char mybuffer[] = "no reason specified";
220             if (sscanf(line, " %[^\n]", word) == -1) {
221                 word = mybuffer;
222             }
223             vt->do_cmd[command](kb, mode, notifynumber, 0, word);
224             continue;
225         }
226         default:;
227         }
228
229         // If a keyboard is inactive, it must be activated before receiving any other commands
230         if(!kb->active){
231             if(command == ACTIVE)
232                 TRY_WITH_RESET(vt->active(kb, mode, notifynumber, 0, 0));
233             continue;
234         }
235         // Specially handled commands only available when keyboard is ACTIVE
236         switch(command){
237         case IDLE:
238             TRY_WITH_RESET(vt->idle(kb, mode, notifynumber, 0, 0));
239             continue;
240         case SWITCH:
241             if(profile->currentmode != mode){
242                 profile->currentmode = mode;
243                 // Set mode light for non-RGB K95
244                 int index = INDEX_OF(mode, profile->mode);
245                 vt->setmodeindex(kb, index);
246             }
247             continue;
248         case HWLOAD: case HWSAVE:{
249             char delay = kb->usbdelay;
250             // Ensure delay of at least 10ms as the device can get overwhelmed otherwise
251             if(delay < 10)
252                 kb->usbdelay = 10;
253             // Try to load/save the hardware profile. Reset on failure, disconnect if reset fails.
254             TRY_WITH_RESET(vt->do_io[command](kb, mode, notifynumber, 1, 0));
255             // Re-send the current RGB state as it sometimes gets scrambled
256             TRY_WITH_RESET(vt->updatergb(kb, 1));
257             kb->usbdelay = delay;
258             continue;
259         }
260         case FWUPDATE:
261             // FW update parses a whole word. Unlike hwload/hwsave, there's no try again on failure.
262             if(vt->fwupdate(kb, mode, notifynumber, 0, word)){
263                 free(word);
264                 return 1;
265             }
266             continue;
267         case POLLRATE: {
268             uint rate;
```

```
269             if(sscanf(word, "%u", &rate) == 1 && (rate == 1 || rate == 2 || rate == 4 || rate == 8))
270                 TRY_WITH_RESET(vt->pollrate(kb, mode, notifynumber, rate, 0));
271             continue;
272         }
273         case ERASEPROFILE:
274             // Erase the current profile
275             vt->eraseprofile(kb, mode, notifynumber, 0, 0);
276             // Update profile/mode pointers
277             profile = kb->profile;
278             mode = profile->currentmode;
279             continue;
280         case ERASE: case NAME: case IOFF: case ION: case IAUTO: case
    INOTIFY: case PROFILENAME: case ID: case PROFILEID: case
    DPISEL: case LIFT: case SNAP:
281             // All of the above just parse the whole word
282             vt->do_cmd[command](kb, mode, notifynumber, 0, word);
283             continue;
284         case RGB: {
285             // RGB command has a special response for a single hex constant
286             int r, g, b;
287             if(sscanf(word, "%02x%02x%02x", &r, &g, &b) == 3){
288                 // Set all keys
289                 for(int i = 0; i < N_KEYS_EXTENDED; i++)
290                     vt->rgb(kb, mode, notifynumber, i, word);
291                 continue;
292             }
293             break;
294         }
295         case MACRO:
296             if(!strcmp(word, "clear")){
297                 // Macro has a special clear command
298                 vt->macro(kb, mode, notifynumber, 0, 0);
299                 continue;
300             }
301             break;
302         default:;
303         }
304         // For anything else, split the parameter at the colon
305         int left = -1;
306         sscanf(word, "%*[^:]%n", &left);
307         if(left <= 0)
308             continue;
309         const char* right = word + left;
310         if(right[0] == ':')
311             right++;
312         // Macros and DPI have a separate left-side handler
313         if(command == MACRO || command == DPI){
314             word[left] = 0;
315             vt->do_macro[command](kb, mode, notifynumber, word, right);
316             continue;
317         }
318         // Scan the left side for key names and run the requested command
319         int position = 0, field = 0;
320         char keyname[11];
321         while(position < left && sscanf(word + position, "%10[^:,]%n", keyname, &field) == 1){
322             int keycode;
323             if(!strcmp(keyname, "all")){
324                 // Set all keys
325                 for(int i = 0; i < N_KEYS_EXTENDED; i++)
326                     vt->do_cmd[command](kb, mode, notifynumber, i, right);
327             } else if((sscanf(keyname, "#%d", &keycode) && keycode >= 0 && keycode <
    N_KEYS_EXTENDED)
328                       || (sscanf(keyname, "#x%x", &keycode) && keycode >= 0 && keycode <
    N_KEYS_EXTENDED)){
329                 // Set a key numerically
330                 vt->do_cmd[command](kb, mode, notifynumber, keycode, right);
331             } else {
332                 // Find this key in the keymap
333                 for(unsigned i = 0; i < N_KEYS_EXTENDED; i++){
334                     if(keymap[i].name && !strcmp(keyname, keymap[i].name)){
335                         vt->do_cmd[command](kb, mode, notifynumber, i, right);
336                         break;
337                     }
338                 }
339             }
340             if(word[position += field] == ',')
341                 position++;
342         }
343     }
344
345     // Finish up
346     if(!NEEDS_FW_UPDATE(kb)){
347         TRY_WITH_RESET(vt->updatergb(kb, 0));
348         TRY_WITH_RESET(vt->updatedpi(kb, 0));
349     }
350     free(word);
351     return 0;
```

```
352 }
```

Here is the call graph for this function:

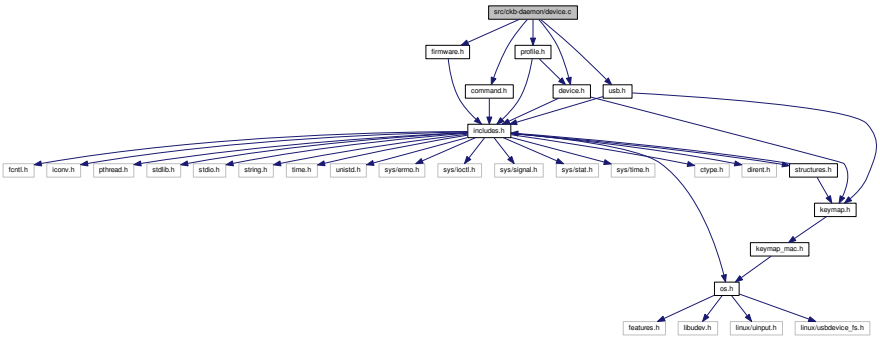


Here is the caller graph for this function:

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

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

Include dependency graph for device.c:



Functions

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

Variables

- int hwload_mode = 1

hwload_mode = 1 means read hardware once. should be enough

- usbdevice keyboard [9]

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

  Mutex for handling the usbdevice structure.
- pthread_mutex_t inputmutex [9] = { [0 ... 9 -1] = PTHREAD_MUTEX_INITIALIZER }

  Mutex for dealing with usb input frames.
- pthread_mutex_t macromutex [9] = { [0 ... 9 -1] = PTHREAD_MUTEX_INITIALIZER }

  Protecting macros against lightning: Both use usb_send.
- pthread_mutex_t macromutex2 [9] = { [0 ... 9 -1] = PTHREAD_MUTEX_INITIALIZER }

  Protecting the single link list of threads and the macrovar.
- pthread_cond_t macrovar [9] = { [0 ... 9 -1] = PTHREAD_COND_INITIALIZER }

  This variable is used to stop and wakeup all macro threads which have to wait.

9.7.1 Function Documentation

9.7.1.1 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 25 of file device.c.

References usbdevice::active, ckb_info, ckb_warn, DEV_MAX, 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().

```
25                                                  {
26      // Get the firmware version from the device
27      if(kb->pollrate == 0){
35          if(!hwload_mode || (HAS_FEATURES(kb, FEAT_HWLOAD) &&
    getfwversion(kb))){
36              if(hwload_mode == 2)
37                  // hwload=always. Report setup failure.
38                  return -1;
39              else if(hwload_mode){
40                  // hwload=once. Log failure, prevent trying again, and continue.
41                  ckb_warn("Unable to load firmware version/poll rate\n");
42                  kb->features &= ~FEAT_HWLOAD;
43              }
44              kb->pollrate = 0;
45              kb->features &= ~(FEAT_POLLRATE | FEAT_ADJRATE);
46              if(kb->fwversion == 0)
47                  kb->features &= ~(FEAT_FWVERSION |
    FEAT_FWUPDATE);
48          }
49      }
54      if(NEEDS_FW_UPDATE(kb)){
56          ckb_info("Device needs a firmware update. Please issue a fwupdate command.\n");
57          kb->features = FEAT_RGB | FEAT_FWVERSION |
    FEAT_FWUPDATE;
58          kb->active = 1;
59          return 0;
60      }
66      if(!kb->hw && hwload_mode && HAS_FEATURES(kb,
    FEAT_HWLOAD)){
67          if(hwloadprofile(kb, 1)){
68              if(hwload_mode == 2)
69                  return -1;
70              ckb_warn("Unable to load hardware profile\n");
71              kb->features &= ~FEAT_HWLOAD;
72          }
73      }
74      // Activate software mode if requested
75      if(makeactive)
76          return setactive(kb, 1);
77      #ifdef DEBUG
78      // 12 for each device + null terminator
79      char devlist[12*(DEV_MAX-1)+1];
80      int devlistpos = 0;
81      for(unsigned i = 1; i < DEV_MAX; i++){
82          devlistpos += sprintf(&devlist[devlistpos], "%u: 0x%x; ", i, keyboard[i].product);
83      }
84      ckb_info("Attached Devices: %s\n", devlist);
85      #endif
86      return 0;
87  }
```

Here is the call graph for this function:

Here is the caller graph for this function:

9.7.1.2 int start_dev (usbdevice * *kb*, int *makeactive*)

Definition at line 89 of file device.c.

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

```
89                                          {
90     // Force USB interval to 10ms during initial setup phase; return to nominal 5ms after setup completes.
91     kb->usbdelay = 10;
92     int res = _start_dev(kb, makeactive);
93     kb->usbdelay = USB_DELAY_DEFAULT;
94     return res;
95 }
```

Here is the call graph for this function:

9.7.2 Variable Documentation

9.7.2.1 pthread_mutex_t devlistmutex = PTHREAD_MUTEX_INITIALIZER

Definition at line 11 of file device.c.

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

Definition at line 12 of file device.c.

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

9.7.2.3 int hwload_mode = 1

hwload_mode is defined in device.c

Definition at line 7 of file device.c.

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

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

Definition at line 13 of file device.c.

9.7.2.5 usbdevice keyboard[9]

Definition at line 10 of file device.c.

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

9.7.2.6 pthread_mutex_t macromutex[9] = { [0 ... 9 -1] = PTHREAD_MUTEX_INITIALIZER }

Definition at line 14 of file device.c.

9.7.2.7 pthread_mutex_t macromutex2[9] = { [0 ... 9 -1] = PTHREAD_MUTEX_INITIALIZER }

Definition at line 15 of file device.c.

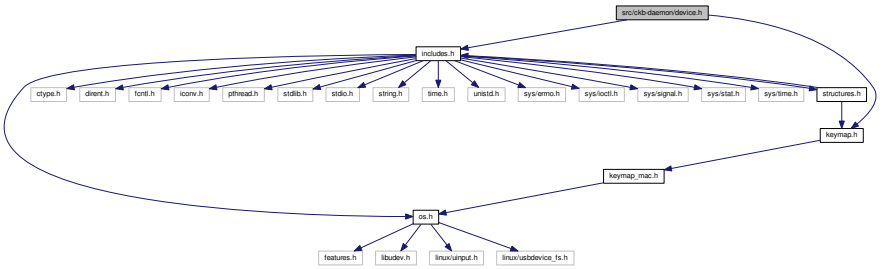
9.7.2.8 pthread_cond_t macrovar[9] = { [0 ... 9 -1] = PTHREAD_COND_INITIALIZER }

Definition at line 16 of file device.c.

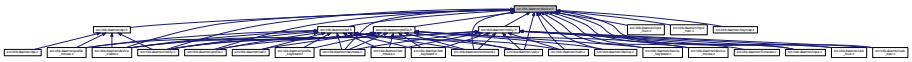
9.8 src/ckb-daemon/device.h File Reference

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

Include dependency graph for device.h:



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



Macros

- #define DEV_MAX 9
- #define IS_CONNECTED(kb) ((kb) && (kb)->handle && (kb)->uinput_kb && (kb)->uinput_mouse)
- #define dmutex(kb) (devmutex + INDEX_OF(kb, keyboard))
- #define imutex(kb) (inputmutex + INDEX_OF(kb, keyboard))
- #define mmutex(kb) (macromutex + INDEX_OF(kb, keyboard))
- #define mmutex2(kb) (macromutex2 + INDEX_OF(kb, keyboard))
- #define mvar(kb) (macrovar + 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]

  remember all usb devices. Needed for closeusb().
- pthread_mutex_t devmutex [9]

  Mutex for handling the usbdevice structure.
- pthread_mutex_t inputmutex [9]

  Mutex for dealing with usb input frames.
- pthread_mutex_t macromutex [9]

  Protecting macros against lightning: Both use usb_send.
- pthread_mutex_t macromutex2 [9]

  Protecting the single link list of threads and the macrovar.
- pthread_cond_t macrovar [9]

  This variable is used to stop and wakeup all macro threads which have to wait.

9.8.1 Macro Definition Documentation

9.8.1.1 #define ACT_LIGHT 1

Definition at line 68 of file device.h.

Referenced by setactive_kb().

9.8.1.2 #define ACT_LOCK 8

Definition at line 71 of file device.h.

Referenced by setactive_kb().

9.8.1.3 #define ACT_M1 10

Definition at line 73 of file device.h.

Referenced by setactive_kb().

9.8.1.4 #define ACT_M2 11

Definition at line 74 of file device.h.

Referenced by setactive_kb().

9.8.1.5 #define ACT_M3 12

Definition at line 75 of file device.h.

Referenced by setactive_kb().

9.8.1.6 #define ACT_MR_RING 9

Definition at line 72 of file device.h.

Referenced by setactive_kb().

9.8.1.7 #define ACT_NEXT 3

Definition at line 69 of file device.h.

9.8.1.8 #define ACT_NEXT_NOWRAP 5

Definition at line 70 of file device.h.

9.8.1.9 #define DEV_MAX 9

Definition at line 8 of file device.h.

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

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

Definition at line 18 of file device.h.

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

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

Definition at line 22 of file device.h.

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

9.8.1.12 #define IN_CORSAIR 0x40

Definition at line 65 of file device.h.

Referenced by setactive_kb(), and setactive_mouse().

9.8.1.13 #define IN_HID 0x80

Definition at line 64 of file device.h.

Referenced by setactive_kb(), and setactive_mouse().

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

Definition at line 12 of file device.h.

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

9.8.1.15 #define mmutex(*kb*) (macromutex + INDEX_OF(kb, keyboard))

Definition at line 26 of file device.h.

Referenced by _usbrecv(), _usbsend(), and play_macro().

9.8.1.16 #define mmutex2(*kb*) (macromutex2 + INDEX_OF(kb, keyboard))

Definition at line 28 of file device.h.

Referenced by play_macro().

9.8.1.17 #define mvar(*kb*) (macrovar + INDEX_OF(kb, keyboard))

Definition at line 30 of file device.h.

Referenced by play_macro().

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

Definition at line 44 of file device.h.

Referenced by _start_dev(), and revertusb().

9.8.2 Function Documentation

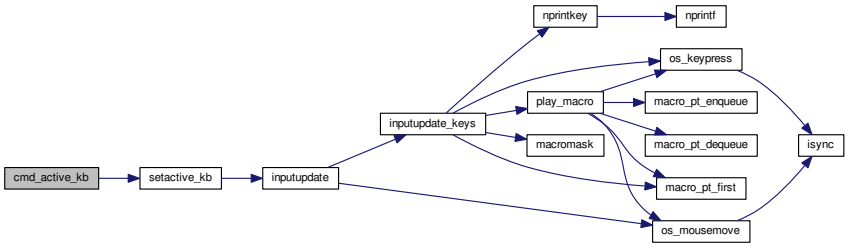
9.8.2.1 int cmd_active_kb (usbdevice * *kb,* usbmode * *dummy1,* int *dummy2,* int *dummy3,* const char * *dummy4*)

Definition at line 114 of file device_keyboard.c.

References setactive_kb().

```
114                                                                                 {
115     (void)dummy1;
116     (void)dummy2;
117     (void)dummy3;
118     (void)dummy4;
119
120     return setactive_kb(kb, 1);
121 }
```

Here is the call graph for this function:



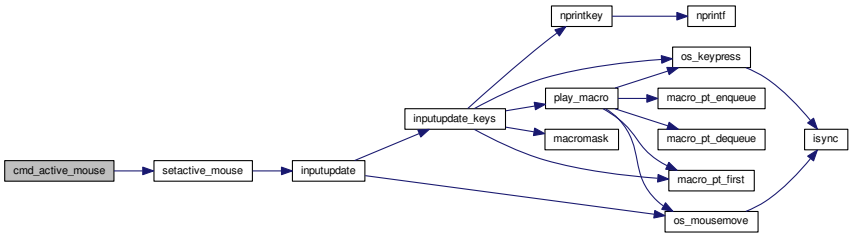
9.8.2.2 int cmd_active_mouse (usbdevice * *kb*, usbmode * *dummy1*, int *dummy2*, int *dummy3*, const char * *dummy4*)

Definition at line 44 of file device_mouse.c.

References setactive_mouse().

```
44                                                                                  {
45     (void)dummy1;
46     (void)dummy2;
47     (void)dummy3;
48     (void)dummy4;
49
50     return setactive_mouse(kb, 1);
51 }
```

Here is the call graph for this function:



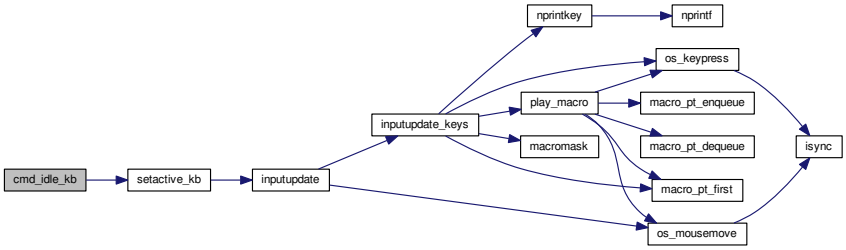
9.8.2.3 int cmd_idle_kb (usbdevice * *kb*, usbmode * *dummy1*, int *dummy2*, int *dummy3*, const char * *dummy4*)

Definition at line 123 of file device_keyboard.c.

References setactive_kb().

```
123                                                                                 {
124     (void)dummy1;
125     (void)dummy2;
126     (void)dummy3;
127     (void)dummy4;
128
129     return setactive_kb(kb, 0);
130 }
```

Here is the call graph for this function:



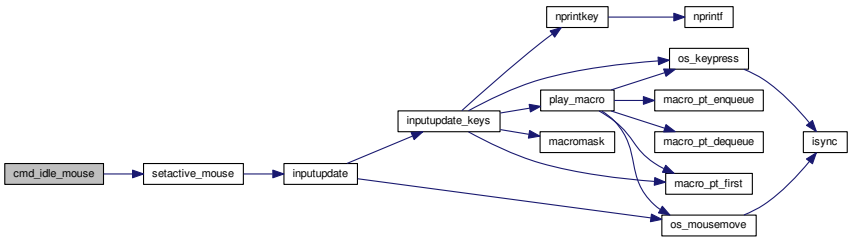
9.8.2.4 int cmd_idle_mouse (usbdevice * *kb*, usbmode * *dummy1*, int *dummy2*, int *dummy3*, const char * *dummy4*)

Definition at line 53 of file device_mouse.c.

References setactive_mouse().

```
53                                                                                                  {
54      (void)dummy1;
55      (void)dummy2;
56      (void)dummy3;
57      (void)dummy4;
58
59      return setactive_mouse(kb, 0);
60  }
```

Here is the call graph for this function:



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

Definition at line 62 of file device_mouse.c.

References MSG_SIZE, usbdevice::pollrate, and usbsend.

```
62                                                                                                  {
63      (void)dummy1;
64      (void)dummy2;
65      (void)dummy3;
66
67      uchar msg[MSG_SIZE] = {
68          0x07, 0x0a, 0, 0, (uchar)rate
69      };
70      if(!usbsend(kb, msg, 1))
71          return -1;
72      // Device should disconnect+reconnect, but update the poll rate field in case it doesn't
73      kb->pollrate = rate;
74      return 0;
75  }
```

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

Definition at line 20 of file device_keyboard.c.

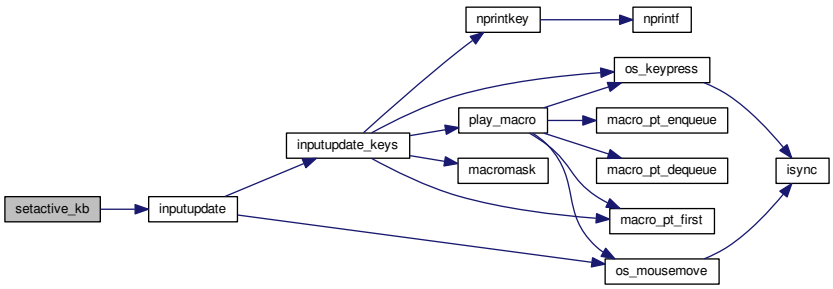
References ACT_LIGHT, ACT_LOCK, ACT_M1, ACT_M2, ACT_M3, ACT_MR_RING, usbdevice::active, DELAY_MEDIUM, lighting::forceupdate, imutex, IN_CORSAIR, IN_HID, usbdevice::input, inputupdate(), keymap, usbinput::keys, usbprofile::lastlight, MSG_SIZE, N_KEYS_HW, NEEDS_FW_UPDATE, usbdevice::profile, usbsend, and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

Referenced by cmd_active_kb(), and cmd_idle_kb().

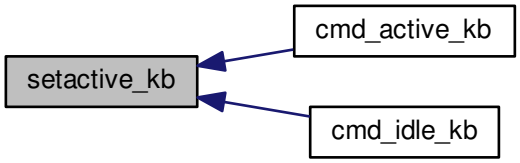
```
20                                          {
21     if(NEEDS_FW_UPDATE(kb))
22         return 0;
23
24     pthread_mutex_lock(imutex(kb));
25     kb->active = !!active;
26     kb->profile->lastlight.forceupdate = 1;
27     // Clear input
28     memset(&kb->input.keys, 0, sizeof(kb->input.keys));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

```
89                      action = ACT_M3;
90                  else if(!strcmp(keymap[key].name, "light"))
91                      action = ACT_LIGHT;
92                  else if(!strcmp(keymap[key].name, "lock"))
93                      action = ACT_LOCK;
94              }
95              msg[1][4 + pair * 2] = key;
96              msg[1][5 + pair * 2] = action;
97          }
98          // Byte 2 = pair count (usually 30, less on final message)
99          msg[1][2] = pair;
100          if(!usbsend(kb, msg[1], 1))
101              return -1;
102      }
103      // Commit new input settings
104      if(!usbsend(kb, msg[2], 1))
105          return -1;
106      DELAY_MEDIUM(kb);
107 #endif
108     }
109     // Update indicator LEDs if the profile contains settings for them
110     kb->vtable->updateindicators(kb, 0);
111     return 0;
112 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



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

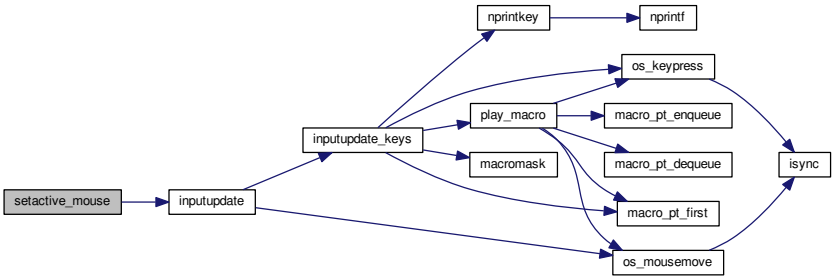
Definition at line 9 of file device_mouse.c.

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

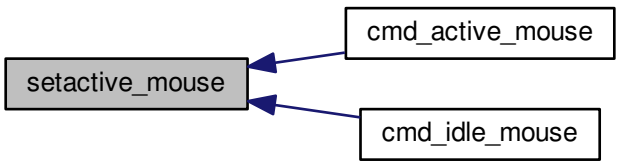
Referenced by cmd_active_mouse(), and cmd_idle_mouse().

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

Here is the call graph for this function:



Here is the caller graph for this function:



9.8.2.8 void setmodeindex_nrgb (usbdevice * kb, int index)

Definition at line 132 of file device_keyboard.c.

References NK95_M1, NK95_M2, NK95_M3, and nk95cmd.

```
132                                                 {
133     switch(index % 3){
134     case 0:
135         nk95cmd(kb, NK95_M1);
136         break;
137     case 1:
138         nk95cmd(kb, NK95_M2);
139         break;
140     case 2:
141         nk95cmd(kb, NK95_M3);
142         break;
143     }
144 }
```

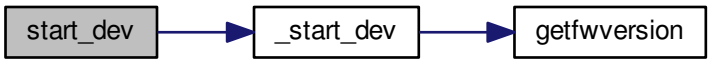
9.8.2.9 int start_dev (usbdevice * kb, int makeactive)

Definition at line 89 of file device.c.

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

```
89                                              {
90     // Force USB interval to 10ms during initial setup phase; return to nominal 5ms after setup completes.
91     kb->usbdelay = 10;
92     int res = _start_dev(kb, makeactive);
93     kb->usbdelay = USB_DELAY_DEFAULT;
94     return res;
95 }
```

Here is the call graph for this function:



9.8.2.10 int start_kb_nrgb (usbdevice * kb, int makeactive)

Definition at line 9 of file device_keyboard.c.

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

```
9                                                   {
10     (void)makeactive;
11
12     // Put the non-RGB K95 into software mode. Nothing else needs to be done hardware wise
13     nk95cmd(kb, NK95_HWOFF);
14     // Fill out RGB features for consistency, even though the keyboard doesn't have them
15     kb->active = 1;
16     kb->pollrate = -1;
17     return 0;
18 }
```

9.8.3 Variable Documentation

9.8.3.1 pthread_mutex_t devmutex[9]

Definition at line 12 of file device.c.

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

9.8.3.2 pthread_mutex_t inputmutex[9]

Definition at line 13 of file device.c.

9.8.3.3 usbdevice keyboard[9]

Definition at line 10 of file device.c.

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

9.8.3.4 pthread_mutex_t macromutex[9]

Definition at line 14 of file device.c.

9.8.3.5 pthread_mutex_t macromutex2[9]

Definition at line 15 of file device.c.

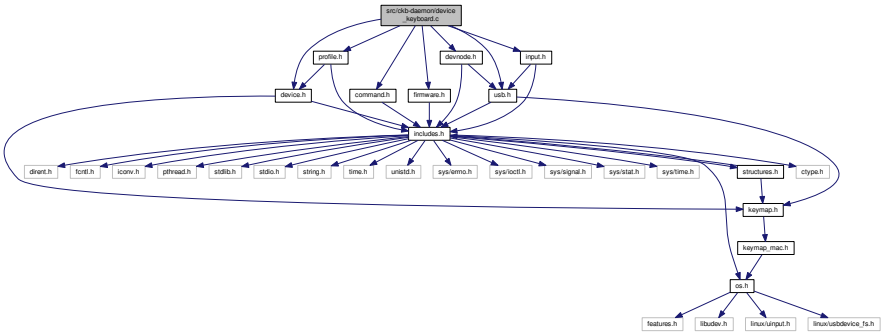
9.8.3.6 pthread_cond_t macrovar[9]

Definition at line 16 of file device.c.

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

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

Include dependency graph for device_keyboard.c:



Functions

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

9.9.1 Function Documentation

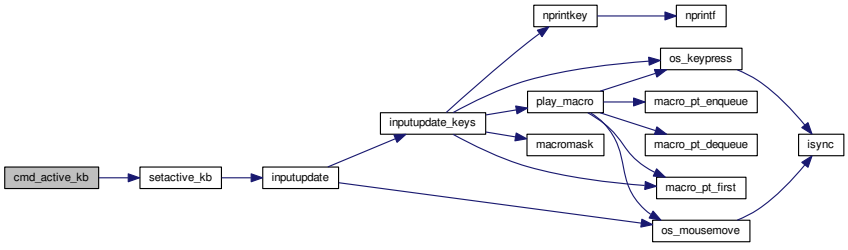
9.9.1.1 int cmd_active_kb (usbdevice * *kb*, usbmode * *dummy1*, int *dummy2*, int *dummy3*, const char * *dummy4*)

Definition at line 114 of file device_keyboard.c.

References setactive_kb().

```
114                                                                                                 {
115     (void)dummy1;
116     (void)dummy2;
117     (void)dummy3;
118     (void)dummy4;
119
120     return setactive_kb(kb, 1);
121 }
```

Here is the call graph for this function:



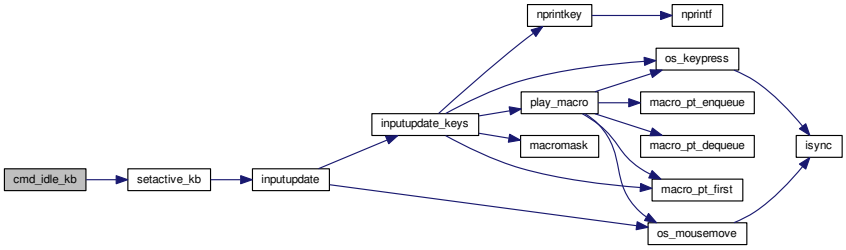
9.9.1.2 int cmd_idle_kb (usbdevice * *kb*, usbmode * *dummy1*, int *dummy2*, int *dummy3*, const char * *dummy4*)

Definition at line 123 of file device_keyboard.c.

References setactive_kb().

```
123                                                                                                 {
124     (void)dummy1;
125     (void)dummy2;
126     (void)dummy3;
127     (void)dummy4;
128
129     return setactive_kb(kb, 0);
130 }
```

Here is the call graph for this function:



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

Definition at line 20 of file device_keyboard.c.

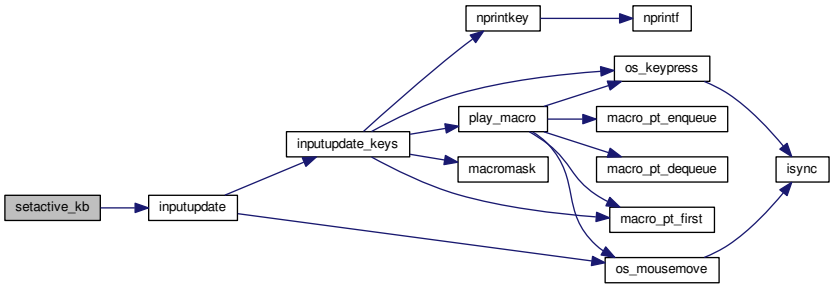
References ACT_LIGHT, ACT_LOCK, ACT_M1, ACT_M2, ACT_M3, ACT_MR_RING, usbdevice::active, DELAY_-MEDIUM, lighting::forceupdate, imutex, IN_CORSAIR, IN_HID, usbdevice::input, inputupdate(), keymap, usbinput-::keys, usbprofile::lastlight, MSG_SIZE, N_KEYS_HW, NEEDS_FW_UPDATE, usbdevice::profile, usbsend,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

Referenced by cmd_active_kb(), and cmd_idle_kb().

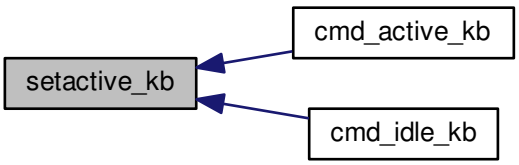
```
20                                          {
21     if(NEEDS_FW_UPDATE(kb))
22         return 0;
23
24     pthread_mutex_lock(imutex(kb));
25     kb->active = !!active;
26     kb->profile->lastlight.forceupdate = 1;
27     // Clear input
28     memset(&kb->input.keys, 0, sizeof(kb->input.keys));
29     inputupdate(kb);
30     pthread_mutex_unlock(imutex(kb));
31
32     uchar msg[3][MSG_SIZE] = {
33         { 0x07, 0x04, 0 },                  // Disables or enables HW control for top row
34         { 0x07, 0x40, 0 },                  // Selects key input
35         { 0x07, 0x05, 2, 0, 0x03, 0x00 }    // Commits key input selection
36     };
37     if(active){
38         // Put the M-keys (K95) as well as the Brightness/Lock keys into software-controlled mode.
39         msg[0][2] = 2;
40         if(!usbsend(kb, msg[0], 1))
41             return -1;
42         DELAY_MEDIUM(kb);
43         // Set input mode on the keys. They must be grouped into packets of 60 bytes (+ 4 bytes header)
44         // Keys are referenced in byte pairs, with the first byte representing the key and the second byte
       representing the mode.
45         for(int key = 0; key < N_KEYS_HW; ){
46             int pair;
47             for(pair = 0; pair < 30 && key < N_KEYS_HW; pair++, key++){
48                 // Select both standard and Corsair input. The standard input will be ignored except in
```

```
    BIOS mode.
49                  uchar action = IN_HID | IN_CORSAIR;
50                  // Additionally, make MR activate the MR ring (this is disabled for now, may be back later)
51                  //if(keymap[key].name && !strcmp(keymap[key].name, "mr"))
52                  //    action |= ACT_MR_RING;
53                  msg[1][4 + pair * 2] = key;
54                  msg[1][5 + pair * 2] = action;
55              }
56              // Byte 2 = pair count (usually 30, less on final message)
57              msg[1][2] = pair;
58              if(!usbsend(kb, msg[1], 1))
59                  return -1;
60          }
61          // Commit new input settings
62          if(!usbsend(kb, msg[2], 1))
63              return -1;
64          DELAY_MEDIUM(kb);
65      } else {
66          // Set the M-keys back into hardware mode, restore hardware RGB profile. It has to be sent twice
    for some reason.
67          msg[0][2] = 1;
68          if(!usbsend(kb, msg[0], 1))
69              return -1;
70          DELAY_MEDIUM(kb);
71          if(!usbsend(kb, msg[0], 1))
72              return -1;
73          DELAY_MEDIUM(kb);
74 #ifdef OS_LINUX
75          // On OSX the default key mappings are fine. On Linux, the G keys will freeze the keyboard. Set the
    keyboard entirely to HID input.
76          for(int key = 0; key < N_KEYS_HW; ){
77              int pair;
78              for(pair = 0; pair < 30 && key < N_KEYS_HW; pair++, key++){
79                  uchar action = IN_HID;
80                  // Enable hardware actions
81                  if(keymap[key].name){
82                      if(!strcmp(keymap[key].name, "mr"))
83                          action = ACT_MR_RING;
84                      else if(!strcmp(keymap[key].name, "m1"))
85                          action = ACT_M1;
86                      else if(!strcmp(keymap[key].name, "m2"))
87                          action = ACT_M2;
88                      else if(!strcmp(keymap[key].name, "m3"))
89                          action = ACT_M3;
90                      else if(!strcmp(keymap[key].name, "light"))
91                          action = ACT_LIGHT;
92                      else if(!strcmp(keymap[key].name, "lock"))
93                          action = ACT_LOCK;
94                  }
95                  msg[1][4 + pair * 2] = key;
96                  msg[1][5 + pair * 2] = action;
97              }
98              // Byte 2 = pair count (usually 30, less on final message)
99              msg[1][2] = pair;
100              if(!usbsend(kb, msg[1], 1))
101                  return -1;
102          }
103          // Commit new input settings
104          if(!usbsend(kb, msg[2], 1))
105              return -1;
106          DELAY_MEDIUM(kb);
107 #endif
108      }
109      // Update indicator LEDs if the profile contains settings for them
110      kb->vtable->updateindicators(kb, 0);
111      return 0;
112 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



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

Definition at line 132 of file device_keyboard.c.

References NK95_M1, NK95_M2, NK95_M3, and nk95cmd.

```
132                                                 {
133     switch(index % 3){
134     case 0:
135         nk95cmd(kb, NK95_M1);
136         break;
137     case 1:
138         nk95cmd(kb, NK95_M2);
139         break;
140     case 2:
141         nk95cmd(kb, NK95_M3);
142         break;
143     }
144 }
```

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

Definition at line 9 of file device_keyboard.c.

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

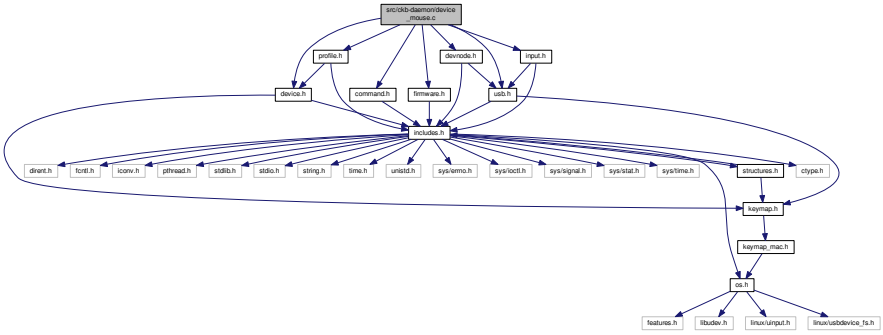
```
9                                                   {
10     (void)makeactive;
11
```

```
12     // Put the non-RGB K95 into software mode. Nothing else needs to be done hardware wise
13     nk95cmd(kb, NK95_HWOFF);
14     // Fill out RGB features for consistency, even though the keyboard doesn't have them
15     kb->active = 1;
16     kb->pollrate = -1;
17     return 0;
18 }
```

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

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

Include dependency graph for device_mouse.c:



Functions

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

9.10.1 Function Documentation

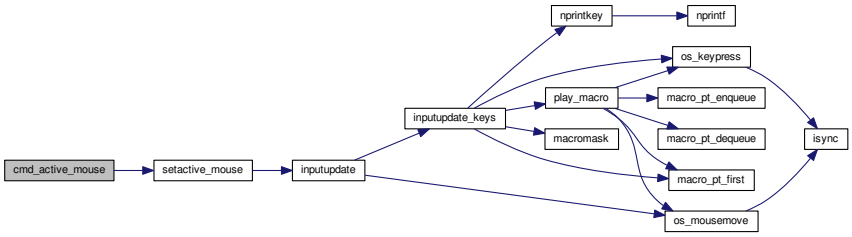
9.10.1.1 int cmd_active_mouse (usbdevice ∗ *kb*, usbmode ∗ *dummy1*, int *dummy2*, int *dummy3*, const char ∗ *dummy4*)

Definition at line 44 of file device_mouse.c.

References setactive_mouse().

```
44                                                                                  {
45     (void)dummy1;
46     (void)dummy2;
47     (void)dummy3;
48     (void)dummy4;
49
50     return setactive_mouse(kb, 1);
51 }
```

Here is the call graph for this function:



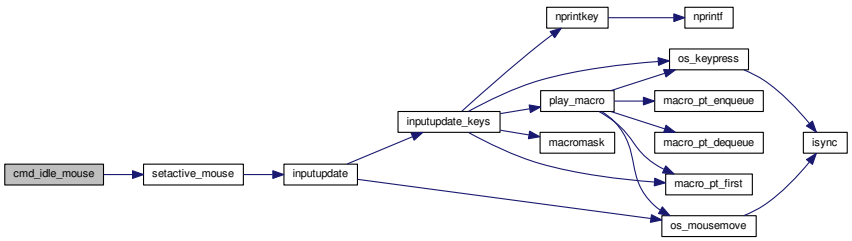
9.10.1.2 int cmd_idle_mouse (usbdevice * *kb*, usbmode * *dummy1*, int *dummy2*, int *dummy3*, const char * *dummy4*)

Definition at line 53 of file device_mouse.c.

References setactive_mouse().

```
53                                                                                                  {
54      (void)dummy1;
55      (void)dummy2;
56      (void)dummy3;
57      (void)dummy4;
58
59      return setactive_mouse(kb, 0);
60  }
```

Here is the call graph for this function:



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

Definition at line 62 of file device_mouse.c.

References MSG_SIZE, usbdevice::pollrate, and usbsend.

```
62                                                                                                  {
63      (void)dummy1;
64      (void)dummy2;
65      (void)dummy3;
66
67      uchar msg[MSG_SIZE] = {
68          0x07, 0x0a, 0, 0, (uchar)rate
69      };
70      if(!usbsend(kb, msg, 1))
71          return -1;
72      // Device should disconnect+reconnect, but update the poll rate field in case it doesn't
73      kb->pollrate = rate;
74      return 0;
75  }
```

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

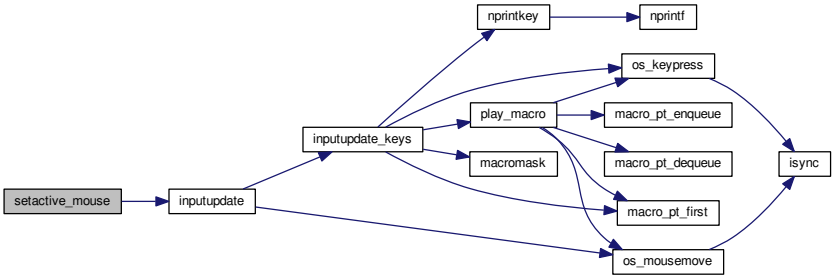
Definition at line 9 of file device_mouse.c.

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

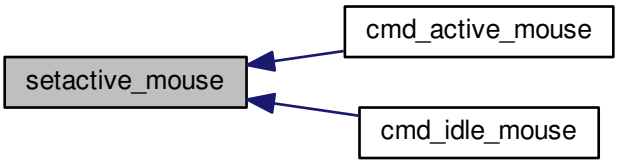
Referenced by cmd_active_mouse(), and cmd_idle_mouse().

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

Here is the call graph for this function:



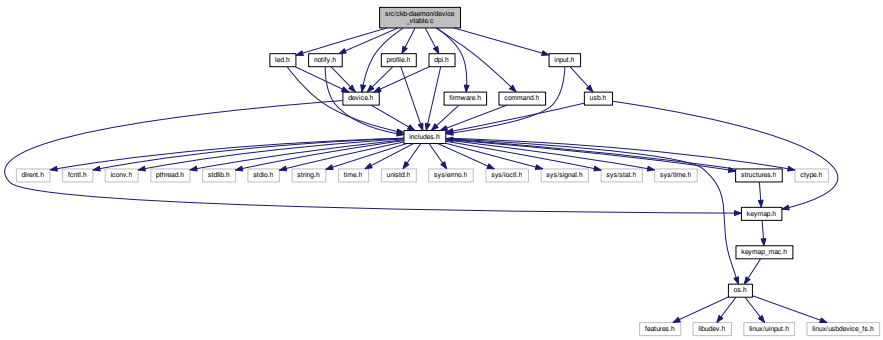
Here is the caller graph for this function:



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

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

Include dependency graph for device_vtable.c:



Functions

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

Variables

- const devcmd vtable_keyboard

RGB keyboard vtable holds functions for each device type.

- const devcmd vtable_keyboard_nonrgb
- const devcmd vtable_mouse

9.11.1 Function Documentation

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

Definition at line 18 of file device_vtable.c.

```
18                                                                                                  {
19     (void)kb;
20     (void)dummy1;
21     (void)dummy2;
22     (void)dummy3;
23     (void)dummy4;
24
25     return 0;
26 }
```

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

Definition at line 27 of file device_vtable.c.

```
27
        {
28     (void)kb;
29     (void)dummy1;
30     (void)dummy2;
31     (void)dummy3;
32     (void)dummy4;
33 }
```

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

Definition at line 11 of file device_vtable.c.

```
11                                                                                                  {
12     (void)kb;
13     (void)dummy1;
14     (void)dummy2;
15     (void)dummy3;
16     (void)dummy4;
17 }
```

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

Definition at line 43 of file device_vtable.c.

```
43                                                  {
44     (void)kb;
45     (void)dummy;
46
47     return 0;
48 }
```

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

Definition at line 39 of file device_vtable.c.

```
39                                                      {
40     (void)kb;
41     (void)dummy;
42 }
```

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

Definition at line 34 of file device_vtable.c.

```
34                                              {
35     (void)kb;
36
37     return 0;
38 }
```

9.11.2 Variable Documentation

9.11.2.1 const devcmd vtable_keyboard

Definition at line 52 of file device_vtable.c.

Referenced by get_vtable().

9.11.2.2 const devcmd vtable_keyboard_nonrgb

Definition at line 99 of file device_vtable.c.

Referenced by get_vtable().

9.11.2.3 const devcmd vtable_mouse

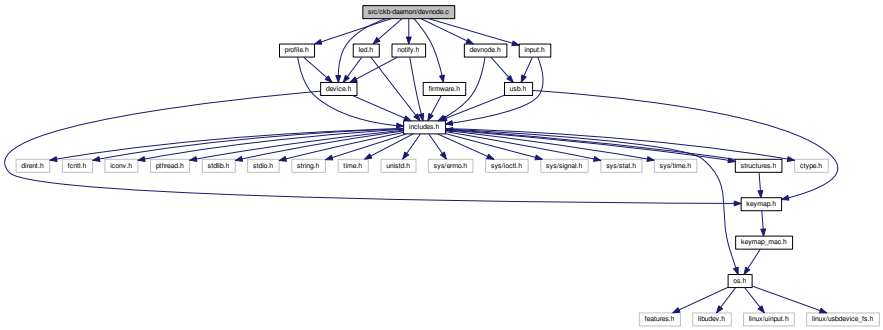
Definition at line 146 of file device_vtable.c.

Referenced by get_vtable().

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

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

Include dependency graph for devnode.c:



Data Structures

- struct _readlines_ctx

Macros

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

Functions

- int rm_recursive (const char *path)
- void _updateconnected ()

  _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

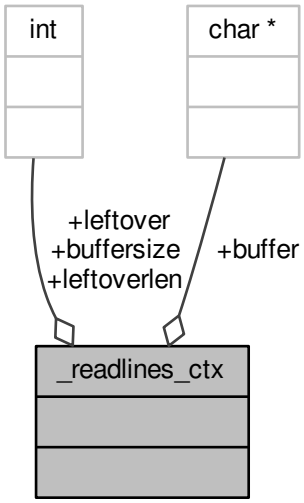
  Group ID for the control nodes. -1 to give read/write access to everybody.

9.12.1 Data Structure Documentation

9.12.1.1 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	

9.12.2 Macro Definition Documentation

9.12.2.1 #define MAX_BUFFER (1024 * 1024 - 1)

Definition at line 334 of file devnode.c.

Referenced by readlines().

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

Definition at line 17 of file devnode.c.

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

9.12.3 Function Documentation

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

Definition at line 136 of file devnode.c.

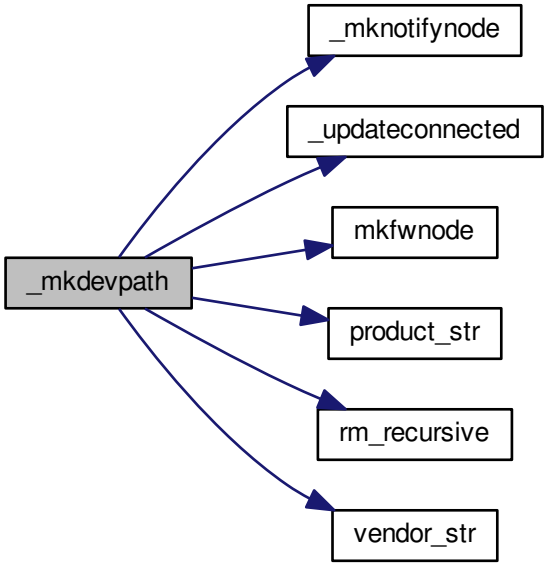
References _mknotifynode(), _updateconnected(), ckb_err, ckb_warn, devpath, FEAT_ADJRATE, FEAT_BIND, FEAT_FWUPDATE, FEAT_FWVERSION, FEAT_MONOCHROME, FEAT_NOTIFY, FEAT_POLLRATE, FEAT_RGB, gid, HAS_FEATURES, INDEX_OF, usbdevice::infifo, keyboard, mkfwnode(), usbdevice::name, usbdevice::product, product_str(), 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
190                 || (kb->infifo = open(inpath, O_RDWR) + 1) == 0){
191             // Add one to the FD because 0 is a valid descriptor, but ckb uses 0 for uninitialized devices
192             ckb_err("Unable to create %s: %s\n", inpath, strerror(errno));
193             rm_recursive(path);
194             kb->infifo = 0;
195             return -1;
196         }
197         if(gid >= 0)
198             fchown(kb->infifo - 1, 0, gid);
199 
200         // Create notification FIFO
201         _mknotifynode(kb, 0);
202 
203         // Write the model and serial to files
204         char mpath[sizeof(path) + 6], spath[sizeof(path) + 7];
205         snprintf(mpath, sizeof(mpath), "%s/model", path);
206         snprintf(spath, sizeof(spath), "%s/serial", path);
```

```
207         FILE* mfile = fopen(mpath, "w");
208         if(mfile){
209             fputs(kb->name, mfile);
210             fputc('\n', mfile);
211             fclose(mfile);
212             chmod(mpath, S_GID_READ);
213             if(gid >= 0)
214                 chown(mpath, 0, gid);
215         } else {
216             ckb_warn("Unable to create %s: %s\n", mpath, strerror(errno));
217             remove(mpath);
218         }
219         FILE* sfile = fopen(spath, "w");
220         if(sfile){
221             fputs(kb->serial, sfile);
222             fputc('\n', sfile);
223             fclose(sfile);
224             chmod(spath, S_GID_READ);
225             if(gid >= 0)
226                 chown(spath, 0, gid);
227         } else {
228             ckb_warn("Unable to create %s: %s\n", spath, strerror(errno));
229             remove(spath);
230         }
231         // Write the keyboard's features
232         char fpath[sizeof(path) + 9];
233         snprintf(fpath, sizeof(fpath), "%s/features", path);
234         FILE* ffile = fopen(fpath, "w");
235         if(ffile){
236             fprintf(ffile, "%s %s", vendor_str(kb->vendor),
    product_str(kb->product));
237             if(HAS_FEATURES(kb, FEAT_MONOCHROME))
238                 fputs(" monochrome", ffile);
239             if(HAS_FEATURES(kb, FEAT_RGB))
240                 fputs(" rgb", ffile);
241             if(HAS_FEATURES(kb, FEAT_POLLRATE))
242                 fputs(" pollrate", ffile);
243             if(HAS_FEATURES(kb, FEAT_ADJRATE))
244                 fputs(" adjrate", ffile);
245             if(HAS_FEATURES(kb, FEAT_BIND))
246                 fputs(" bind", ffile);
247             if(HAS_FEATURES(kb, FEAT_NOTIFY))
248                 fputs(" notify", ffile);
249             if(HAS_FEATURES(kb, FEAT_FWVERSION))
250                 fputs(" fwversion", ffile);
251             if(HAS_FEATURES(kb, FEAT_FWUPDATE))
252                 fputs(" fwupdate", ffile);
253             fputc('\n', ffile);
254             fclose(ffile);
255             chmod(fpath, S_GID_READ);
256             if(gid >= 0)
257                 chown(fpath, 0, gid);
258         } else {
259             ckb_warn("Unable to create %s: %s\n", fpath, strerror(errno));
260             remove(fpath);
261         }
262         // Write firmware version and poll rate
263         mkfwnode(kb);
264     }
265     return 0;
266 }
```

Here is the call graph for this function:



Here is the caller graph for this function:

9.12.3.2 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:

9.12.3.3 int _rmnotifynode (usbdevice * kb, int notify)

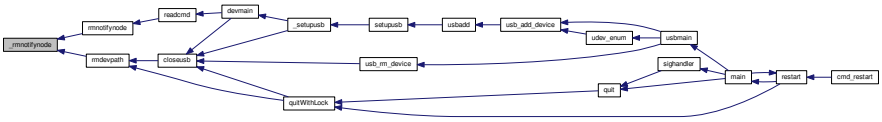
Definition at line 115 of file devnode.c.

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:



9.12.3.4 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 0D02303DBACBACBACBACBACBACBAC998 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.

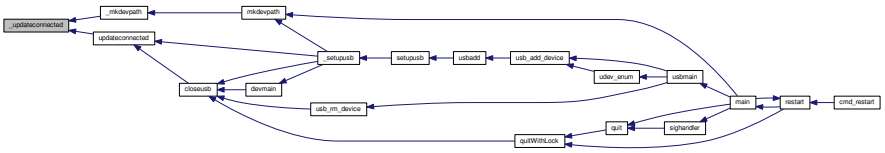
References ckb_warn, DEV_MAX, devmutex, devpath, gid, IS_CONNECTED, keyboard, and S_GID_READ.

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:



9.12.3.5 int mkdevpath (usbdevice * *kb*)

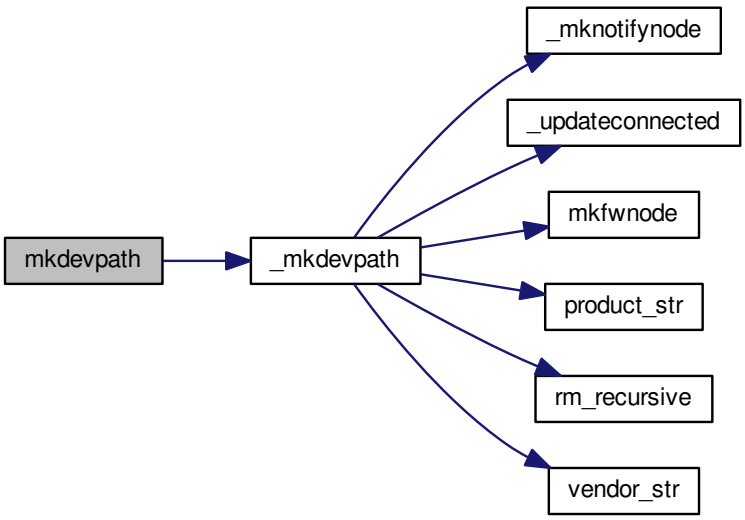
Definition at line 268 of file devnode.c.

References _mkdevpath(), euid_guard_start, and euid_guard_stop.

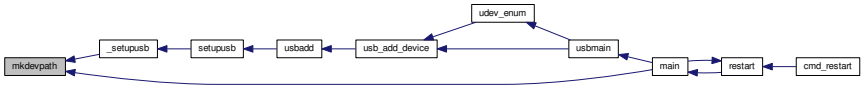
Referenced by _setupusb(), and main().

```
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:



9.12.3.6 int mkfwnode (usbdevice * *kb*)

Definition at line 299 of file devnode.c.

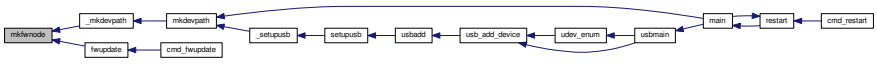
References ckb_warn, devpath, usbdevice::fwversion, gid, INDEX_OF, keyboard, usbdevice::pollrate, and S_GID_-READ.

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");
```

```
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:



9.12.3.7 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().

```
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:

9.12.3.8 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:

9.12.3.9 void readlines_ctx_free (readlines_ctx *ctx*)

Definition at line 348 of file devnode.c.

References _readlines_ctx::buffer.

Referenced by devmain().

```
348                                             {
```

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

Here is the caller graph for this function:

9.12.3.10 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:

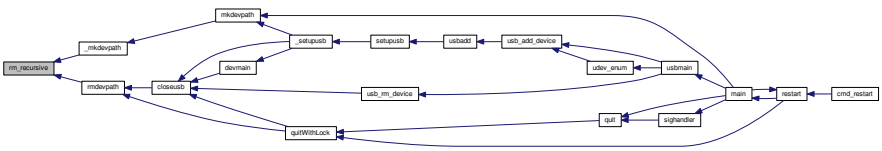
9.12.3.11 int rm_recursive (const char * *path*)

Definition at line 19 of file devnode.c.

Referenced by _mkdevpath(), and rmdevpath().

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

Here is the caller graph for this function:



9.12.3.12 int rmdevpath (usbdevice * *kb*)

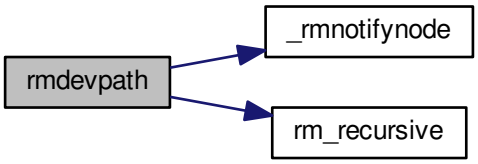
Definition at line 275 of file devnode.c.

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

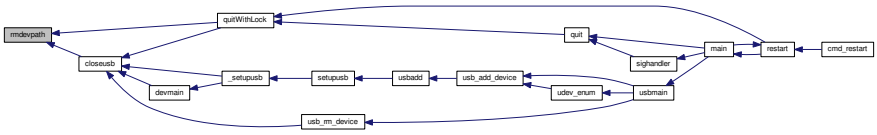
Referenced by closeusb(), and quitWithLock().

```
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:



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

Definition at line 129 of file devnode.c.

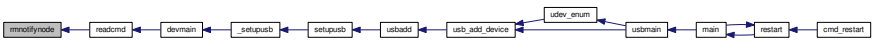
References _rmnotifynode(), euid_guard_start, and euid_guard_stop.

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:



9.12.3.14 void updateconnected ( )

Definition at line 81 of file devnode.c.

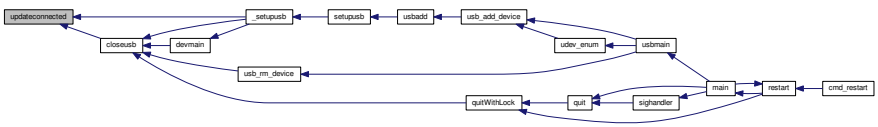
References _updateconnected(), euid_guard_start, and euid_guard_stop.

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:



9.12.4 Variable Documentation

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

Definition at line 11 of file devnode.c.

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

9.12.4.2 long gid = -1

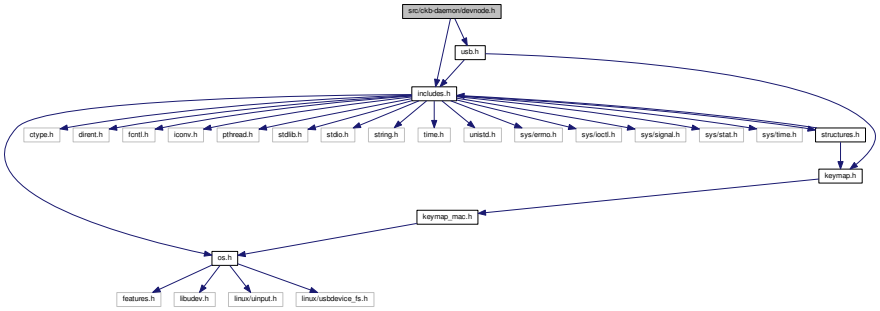
Definition at line 16 of file devnode.c.

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

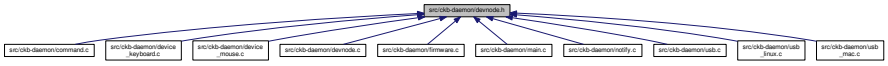
9.13 src/ckb-daemon/devnode.h File Reference

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

Include dependency graph for devnode.h:



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



Macros

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

Typedefs

- typedef struct _readlines_ctx * readlines_ctx

  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.

9.13.1 Macro Definition Documentation

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

Definition at line 17 of file devnode.h.

Referenced by _mkdevpath().

9.13.1.2 #define S_CUSTOM_R (S_IRUSR | S_IWUSR | S_IRGRP)

Definition at line 18 of file devnode.h.

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

Definition at line 15 of file devnode.h.

Referenced by _mkdevpath().

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

Definition at line 14 of file devnode.h.

Referenced by _mkdevpath().

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

Definition at line 16 of file devnode.h.

Referenced by _mkdevpath().

9.13.2 Typedef Documentation

9.13.2.1 typedef struct _readlines_ctx* readlines_ctx

Definition at line 39 of file devnode.h.

9.13.3 Function Documentation

9.13.3.1 int mkdevpath (usbdevice * kb)

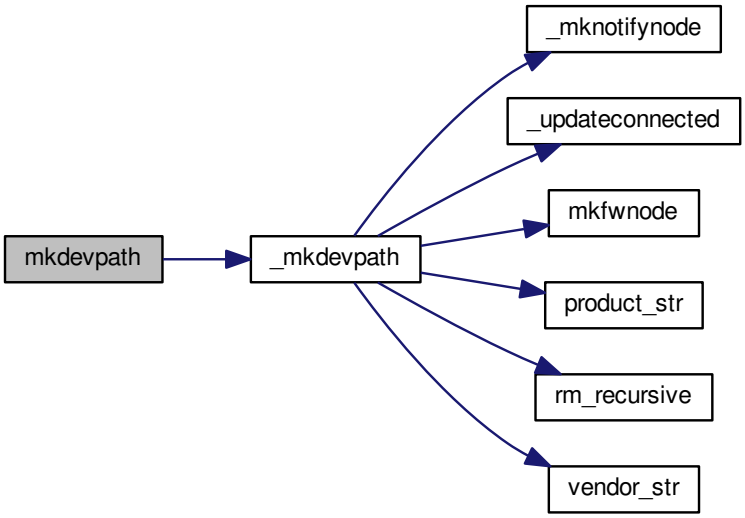
Definition at line 268 of file devnode.c.

References _mkdevpath(), euid_guard_start, and euid_guard_stop.

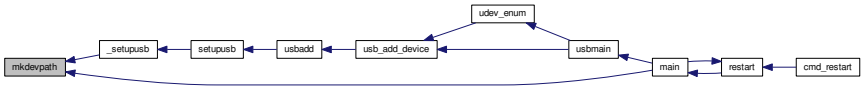
Referenced by _setupusb(), and main().

```
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:



9.13.3.2 int mkfwnode (usbdevice * kb)

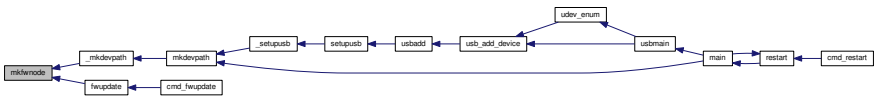
Definition at line 299 of file devnode.c.

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:



9.13.3.3 int mknotifynode (usbdevice * kb, int notify)

Definition at line 108 of file devnode.c.

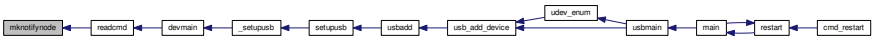
References _mknotifynode(), euid_guard_start, and euid_guard_stop.

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:



9.13.3.4 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().

```
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:

9.13.3.5 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:

9.13.3.6 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:

9.13.3.7 int rmdevpath (usbdevice * kb)

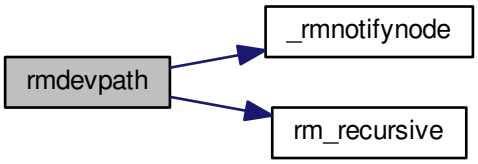
Definition at line 275 of file devnode.c.

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

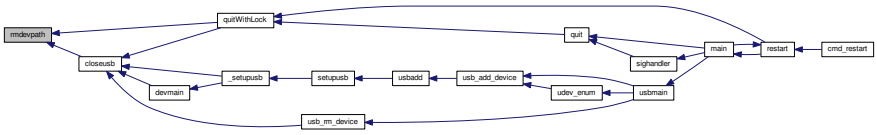
Referenced by closeusb(), and quitWithLock().

```
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:



9.13.3.8 int rmnotifynode (usbdevice * kb, int notify)

Definition at line 129 of file devnode.c.

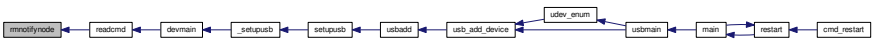
References _rmnotifynode(), euid_guard_start, and euid_guard_stop.

Referenced by readcmd().

```
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:



9.13.3.9 void updateconnected ( )

Definition at line 81 of file devnode.c.

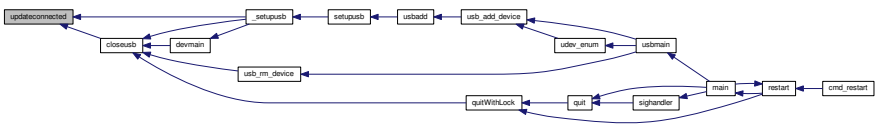
References _updateconnected(), euid_guard_start, and euid_guard_stop.

Referenced by _setupusb(), and closeusb().

```
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:



9.13.4 Variable Documentation

9.13.4.1 const char∗ const devpath

Definition at line 8 of file devnode.h.

9.13.4.2 long gid

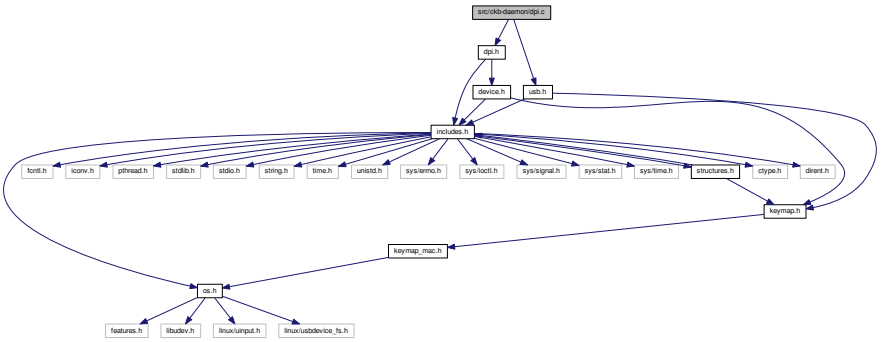
Definition at line 16 of file devnode.c.

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

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

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

Include dependency graph for dpi.c:



Functions

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

9.14.1 Function Documentation

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

Definition at line 4 of file dpi.c.

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

```
4                                                                                    {
5     (void)kb;
6     (void)dummy;
7
8     int disable = 0;
```

```
9       ushort x, y;
10      // Try to scan X,Y values
11      if(sscanf(values, "%hu,%hu", &x, &y) != 2){
12          // If that doesn't work, scan single number
13          if(sscanf(values, "%hu", &x) == 1)
14              y = x;
15          else if(!strncmp(values, "off", 3))
16              // If the right side says "off", disable the level(s)
17              disable = 1;
18          else
19              // Otherwise, quit
20              return;
21      }
22      if((x == 0 || y == 0) && !disable)
23          return;
24      // Scan the left side for stage numbers (comma-separated)
25      int left = strlen(stages);
26      int position = 0, field = 0;
27      char stagename[3];
28      while(position < left && sscanf(stages + position, "%2[^,]%n", stagename, &field) == 1){
29          uchar stagenum;
30          if(sscanf(stagename, "%hhu", &stagenum) && stagenum < DPI_COUNT){
31              // Set DPI for this stage
32              if(disable){
33                  mode->dpi.enabled &= ~(1 << stagenum);
34                  mode->dpi.x[stagenum] = 0;
35                  mode->dpi.y[stagenum] = 0;
36              } else {
37                  mode->dpi.enabled |= 1 << stagenum;
38                  mode->dpi.x[stagenum] = x;
39                  mode->dpi.y[stagenum] = y;
40              }
41          }
42          if(stages[position += field] == ',')
43              position++;
44      }
45  }
```

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

Definition at line 47 of file dpi.c.

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

```
47                                                                                  {
48      (void)kb;
49      (void)dummy1;
50      (void)dummy2;
51
52      uchar stagenum;
53      if(sscanf(stage, "%hhu", &stagenum) != 1)
54          return;
55      if(stagenum > DPI_COUNT)
56          return;
57      mode->dpi.current = stagenum;
58  }
```

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

Definition at line 60 of file dpi.c.

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

```
60                                                                                  {
61      (void)kb;
62      (void)dummy1;
63      (void)dummy2;
64
65      uchar heightnum;
66      if(sscanf(height, "%hhu", &heightnum) != 1)
67          return;
68      if(heightnum > LIFT_MAX || heightnum < LIFT_MIN)
69          return;
70      mode->dpi.lift = heightnum;
71  }
```

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

Definition at line 73 of file dpi.c.

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

```
73                                                                                                  {
74     (void)kb;
75     (void)dummy1;
76     (void)dummy2;
77
78     if(!strcmp(enable, "on"))
79         mode->dpi.snap = 1;
80     if(!strcmp(enable, "off"))
81         mode->dpi.snap = 0;
82 }
```

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

Definition at line 222 of file dpi.c.

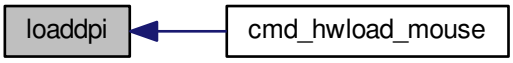
References lighting::b, ckb_err, dpiset::current, DPI_COUNT, dpiset::enabled, lighting::g, LED_MOUSE, dpiset::lift, LIFT_MAX, LIFT_MIN, MSG_SIZE, N_MOUSE_ZONES, lighting::r, dpiset::snap, usbrecv, dpiset::x, and dpiset::y.

Referenced by cmd_hwload_mouse().

```
222                                                                                                 {
223     // Ask for settings
224     uchar data_pkt[4][MSG_SIZE] = {
225         { 0x0e, 0x13, 0x05, 1, },
226         { 0x0e, 0x13, 0x02, 1, },
227         { 0x0e, 0x13, 0x03, 1, },
228         { 0x0e, 0x13, 0x04, 1, }
229     };
230     uchar in_pkt[4][MSG_SIZE];
231     for(int i = 0; i < 4; i++){
232         if(!usbrecv(kb, data_pkt[i], in_pkt[i]))
233             return -2;
234         if(memcmp(in_pkt[i], data_pkt[i], 4)){
235             ckb_err("Bad input header\n");
236             return -3;
237         }
238     }
239     // Copy data from device
240     dpi->enabled = in_pkt[0][4];
241     dpi->enabled &= (1 << DPI_COUNT) - 1;
242     dpi->current = in_pkt[1][4];
243     if(dpi->current >= DPI_COUNT)
244         dpi->current = 0;
245     dpi->lift = in_pkt[2][4];
246     if(dpi->lift < LIFT_MIN || dpi->lift > LIFT_MAX)
247         dpi->lift = LIFT_MIN;
248     dpi->snap = !!in_pkt[3][4];
249
250     // Get X/Y DPIs
251     for(int i = 0; i < DPI_COUNT; i++){
252         uchar data_pkt[MSG_SIZE] = { 0x0e, 0x13, 0xd0, 1 };
253         uchar in_pkt[MSG_SIZE];
254         data_pkt[2] |= i;
255         if(!usbrecv(kb, data_pkt, in_pkt))
256             return -2;
257         if(memcmp(in_pkt, data_pkt, 4)){
258             ckb_err("Bad input header\n");
259             return -3;
260         }
261         // Copy to profile
262         dpi->x[i] = *(ushort*)(in_pkt + 5);
263         dpi->y[i] = *(ushort*)(in_pkt + 7);
264         light->r[LED_MOUSE + N_MOUSE_ZONES + i] = in_pkt[9];
265         light->g[LED_MOUSE + N_MOUSE_ZONES + i] = in_pkt[10];
266         light->b[LED_MOUSE + N_MOUSE_ZONES + i] = in_pkt[11];
267     }
268     // Finished. Set SW DPI light to the current hardware level
269     light->r[LED_MOUSE + 2] = light->r[LED_MOUSE +
    N_MOUSE_ZONES + dpi->current];
270     light->g[LED_MOUSE + 2] = light->g[LED_MOUSE +
    N_MOUSE_ZONES + dpi->current];
271     light->b[LED_MOUSE + 2] = light->b[LED_MOUSE +
    N_MOUSE_ZONES + dpi->current];
```

```
272     return 0;
273 }
```

Here is the caller graph for this function:



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

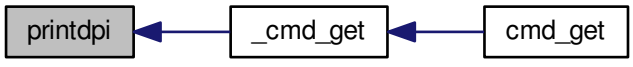
Definition at line 84 of file dpi.c.

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

Referenced by _cmd_get().

```
84                                                  {
85     (void)kb;
86
87     // Print all DPI settings
88     const int BUFFER_LEN = 100;
89     char* buffer = malloc(BUFFER_LEN);
90     int length = 0;
91     for(int i = 0; i < DPI_COUNT; i++){
92         // Print the stage number
93         int newlen = 0;
94         snprintf(buffer + length, BUFFER_LEN - length, length == 0 ? "%d%n" : " %d%n", i, &newlen);
95         length += newlen;
96         // Print the DPI settings
97         if(!(dpi->enabled & (1 << i)))
98             snprintf(buffer + length, BUFFER_LEN - length, ":off%n", &newlen);
99         else
100             snprintf(buffer + length, BUFFER_LEN - length, ":%u,%u%n", dpi->x[i], dpi->
      y[i], &newlen);
101         length += newlen;
102     }
103     return buffer;
104 }
```

Here is the caller graph for this function:



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

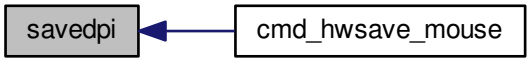
Definition at line 194 of file dpi.c.

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().

```
194                                                     {
195     // Send X/Y DPIs
196     for(int i = 0; i < DPI_COUNT; i++){
197         uchar data_pkt[MSG_SIZE] = { 0x07, 0x13, 0xd0, 1 };
198         data_pkt[2] |= i;
199         *(ushort*)(data_pkt + 5) = dpi->x[i];
200         *(ushort*)(data_pkt + 7) = dpi->y[i];
201         // Save the RGB value for this setting too
202         data_pkt[9] = light->r[LED_MOUSE + N_MOUSE_ZONES + i];
203         data_pkt[10] = light->g[LED_MOUSE + N_MOUSE_ZONES + i];
204         data_pkt[11] = light->b[LED_MOUSE + N_MOUSE_ZONES + i];
205         if(!usbsend(kb, data_pkt, 1))
206             return -1;
207     }
208
209     // Send settings
210     uchar data_pkt[4][MSG_SIZE] = {
211         { 0x07, 0x13, 0x05, 1, dpi->enabled },
212         { 0x07, 0x13, 0x02, 1, dpi->current },
213         { 0x07, 0x13, 0x03, 1, dpi->lift },
214         { 0x07, 0x13, 0x04, 1, dpi->snap, 0x05 }
215     };
216     if(!usbsend(kb, data_pkt[0], 4))
217         return -2;
218     // Finished
219     return 0;
220 }
```

Here is the caller graph for this function:



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

Definition at line 106 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.

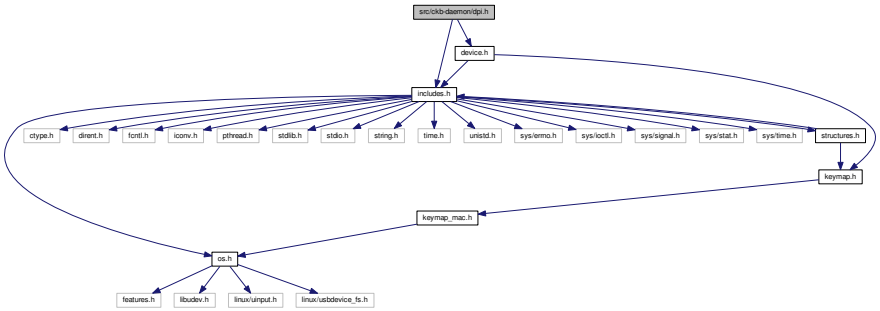
```
106                                         {
107     if(!kb->active)
108         return 0;
109     dpiset* lastdpi = &kb->profile->lastdpi;
110     dpiset* newdpi = &kb->profile->currentmode->dpi;
111     // Don't do anything if the settings haven't changed
112     if(!force && !lastdpi->forceupdate && !newdpi->forceupdate
113             && !memcmp(lastdpi, newdpi, sizeof(dpiset)))
114         return 0;
115     lastdpi->forceupdate = newdpi->forceupdate = 0;
116
117     if (newdpi->current != lastdpi->current) {
118         // Before we switch the current DPI stage, make sure the stage we are
119         // switching to is both enabled and configured to the correct DPI.
120
121         // Enable the stage if necessary.
122         if ((lastdpi->enabled & 1 << newdpi->current) == 0) {
123             uchar newenabled;
124             // If the new enabled flags contain both the current and previous
```

```
125             // stages, use it.
126             if (newdpi->enabled & 1 << newdpi->current &&
127                 newdpi->enabled & 1 << lastdpi->current) {
128                 newenabled = newdpi->enabled;
129             } else {
130                 // Otherwise just enable the new stage. We'll write the actual
131                 // requested flags after switching stages.
132                 newenabled = lastdpi->enabled | 1 << newdpi->current;
133             }
134             uchar data_pkt[MSG_SIZE] = { 0x07, 0x13, 0x05, 0, newenabled };
135             if(!usbsend(kb, data_pkt, 1))
136                 return -2;
137             // Cache the flags we wrote.
138             lastdpi->enabled = newenabled;
139         }
140         // Set the DPI for the new stage if necessary.
141         if (newdpi->x[newdpi->current] != lastdpi->x[newdpi->current] ||
142             newdpi->y[newdpi->current] != lastdpi->y[newdpi->current]) {
143             uchar data_pkt[MSG_SIZE] = { 0x07, 0x13, 0xd0, 0 };
144             data_pkt[2] |= newdpi->current;
145             *(ushort*)(data_pkt + 5) = newdpi->x[newdpi->current];
146             *(ushort*)(data_pkt + 7) = newdpi->y[newdpi->current];
147             if(!usbsend(kb, data_pkt, 1))
148                 return -1;
149             // Set these values in the cache so we don't rewrite them.
150             lastdpi->x[newdpi->current] = newdpi->x[newdpi->current];
151             lastdpi->y[newdpi->current] = newdpi->y[newdpi->current];
152         }
153         // Set current DPI stage.
154         uchar data_pkt[MSG_SIZE] = { 0x07, 0x13, 0x02, 0, newdpi->
    current };
155         if(!usbsend(kb, data_pkt, 1))
156             return -2;
157     }
158
159     // Send X/Y DPIs. We've changed to the new stage already so these can be set
160     // safely.
161     for(int i = 0; i < DPI_COUNT; i++){
162         if (newdpi->x[i] == lastdpi->x[i] && newdpi->y[i] == lastdpi->y[i])
163             continue;
164         uchar data_pkt[MSG_SIZE] = { 0x07, 0x13, 0xd0, 0 };
165         data_pkt[2] |= i;
166         *(ushort*)(data_pkt + 5) = newdpi->x[i];
167         *(ushort*)(data_pkt + 7) = newdpi->y[i];
168         if(!usbsend(kb, data_pkt, 1))
169             return -1;
170     }
171
172     // Send settings
173     if (newdpi->enabled != lastdpi->enabled) {
174         uchar data_pkt[MSG_SIZE] = { 0x07, 0x13, 0x05, 0, newdpi->
    enabled };
175         if(!usbsend(kb, data_pkt, 1))
176             return -2;
177     }
178     if (newdpi->lift != lastdpi->lift) {
179         uchar data_pkt[MSG_SIZE] = { 0x07, 0x13, 0x03, 0, newdpi->
    lift };
180         if(!usbsend(kb, data_pkt, 1))
181             return -2;
182     }
183     if (newdpi->snap != lastdpi->snap) {
184         uchar data_pkt[MSG_SIZE] = { 0x07, 0x13, 0x04, 0, newdpi->
    snap, 0x05 };
185         if(!usbsend(kb, data_pkt, 1))
186             return -2;
187     }
188
189     // Finished
190     memcpy(lastdpi, newdpi, sizeof(dpiset));
191     return 0;
192 }
```

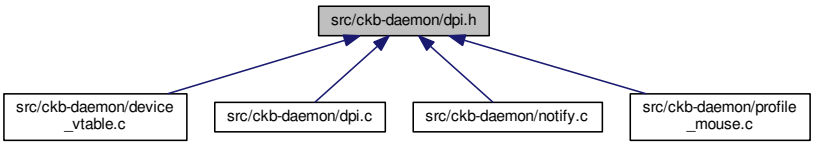
9.15 src/ckb-daemon/dpi.h File Reference

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

Include dependency graph for dpi.h:



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



Functions

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

9.15.1 Function Documentation

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

Definition at line 4 of file dpi.c.

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

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

```
18          else
19              // Otherwise, quit
20              return;
21      }
22      if((x == 0 || y == 0) && !disable)
23          return;
24      // Scan the left side for stage numbers (comma-separated)
25      int left = strlen(stages);
26      int position = 0, field = 0;
27      char stagename[3];
28      while(position < left && sscanf(stages + position, "%2[^,]%n", stagename, &field) == 1){
29          uchar stagenum;
30          if(sscanf(stagename, "%hhu", &stagenum) && stagenum < DPI_COUNT){
31              // Set DPI for this stage
32              if(disable){
33                  mode->dpi.enabled &= ~(1 << stagenum);
34                  mode->dpi.x[stagenum] = 0;
35                  mode->dpi.y[stagenum] = 0;
36              } else {
37                  mode->dpi.enabled |= 1 << stagenum;
38                  mode->dpi.x[stagenum] = x;
39                  mode->dpi.y[stagenum] = y;
40              }
41          }
42          if(stages[position += field] == ',')
43              position++;
44      }
45  }
```

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

Definition at line 47 of file dpi.c.

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

```
47                                                                                  {
48      (void)kb;
49      (void)dummy1;
50      (void)dummy2;
51
52      uchar stagenum;
53      if(sscanf(stage, "%hhu", &stagenum) != 1)
54          return;
55      if(stagenum > DPI_COUNT)
56          return;
57      mode->dpi.current = stagenum;
58  }
```

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

Definition at line 60 of file dpi.c.

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

```
60                                                                                  {
61      (void)kb;
62      (void)dummy1;
63      (void)dummy2;
64
65      uchar heightnum;
66      if(sscanf(height, "%hhu", &heightnum) != 1)
67          return;
68      if(heightnum > LIFT_MAX || heightnum < LIFT_MIN)
69          return;
70      mode->dpi.lift = heightnum;
71  }
```

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

Definition at line 73 of file dpi.c.

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

```
73                                                                                  {
74      (void)kb;
75      (void)dummy1;
76      (void)dummy2;
77
78      if(!strcmp(enable, "on"))
79          mode->dpi.snap = 1;
80      if(!strcmp(enable, "off"))
81          mode->dpi.snap = 0;
82  }
```

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

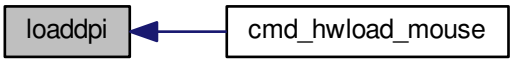
Definition at line 222 of file dpi.c.

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().

```
222                                                                                 {
223     // Ask for settings
224     uchar data_pkt[4][MSG_SIZE] = {
225         { 0x0e, 0x13, 0x05, 1, },
226         { 0x0e, 0x13, 0x02, 1, },
227         { 0x0e, 0x13, 0x03, 1, },
228         { 0x0e, 0x13, 0x04, 1, }
229     };
230     uchar in_pkt[4][MSG_SIZE];
231     for(int i = 0; i < 4; i++){
232         if(!usbrecv(kb, data_pkt[i], in_pkt[i]))
233             return -2;
234         if(memcmp(in_pkt[i], data_pkt[i], 4)){
235             ckb_err("Bad input header\n");
236             return -3;
237         }
238     }
239     // Copy data from device
240     dpi->enabled = in_pkt[0][4];
241     dpi->enabled &= (1 << DPI_COUNT) - 1;
242     dpi->current = in_pkt[1][4];
243     if(dpi->current >= DPI_COUNT)
244         dpi->current = 0;
245     dpi->lift = in_pkt[2][4];
246     if(dpi->lift < LIFT_MIN || dpi->lift > LIFT_MAX)
247         dpi->lift = LIFT_MIN;
248     dpi->snap = !!in_pkt[3][4];
249
250     // Get X/Y DPIs
251     for(int i = 0; i < DPI_COUNT; i++){
252         uchar data_pkt[MSG_SIZE] = { 0x0e, 0x13, 0xd0, 1 };
253         uchar in_pkt[MSG_SIZE];
254         data_pkt[2] |= i;
255         if(!usbrecv(kb, data_pkt, in_pkt))
256             return -2;
257         if(memcmp(in_pkt, data_pkt, 4)){
258             ckb_err("Bad input header\n");
259             return -3;
260         }
261         // Copy to profile
262         dpi->x[i] = *(ushort*)(in_pkt + 5);
263         dpi->y[i] = *(ushort*)(in_pkt + 7);
264         light->r[LED_MOUSE + N_MOUSE_ZONES + i] = in_pkt[9];
265         light->g[LED_MOUSE + N_MOUSE_ZONES + i] = in_pkt[10];
266         light->b[LED_MOUSE + N_MOUSE_ZONES + i] = in_pkt[11];
267     }
268     // Finished. Set SW DPI light to the current hardware level
269     light->r[LED_MOUSE + 2] = light->r[LED_MOUSE +
      N_MOUSE_ZONES + dpi->current];
270     light->g[LED_MOUSE + 2] = light->g[LED_MOUSE +
      N_MOUSE_ZONES + dpi->current];
271     light->b[LED_MOUSE + 2] = light->b[LED_MOUSE +
      N_MOUSE_ZONES + dpi->current];
272     return 0;
273 }
```

Here is the caller graph for this function:



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

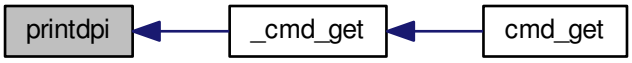
Definition at line 84 of file dpi.c.

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

Referenced by _cmd_get().

```
84                                                      {
85     (void)kb;
86
87     // Print all DPI settings
88     const int BUFFER_LEN = 100;
89     char* buffer = malloc(BUFFER_LEN);
90     int length = 0;
91     for(int i = 0; i < DPI_COUNT; i++){
92         // Print the stage number
93         int newlen = 0;
94         snprintf(buffer + length, BUFFER_LEN - length, length == 0 ? "%d%n" : " %d%n", i, &newlen);
95         length += newlen;
96         // Print the DPI settings
97         if(!(dpi->enabled & (1 << i)))
98             snprintf(buffer + length, BUFFER_LEN - length, ":off%n", &newlen);
99         else
100             snprintf(buffer + length, BUFFER_LEN - length, ":%u,%u%n", dpi->x[i], dpi->
      y[i], &newlen);
101         length += newlen;
102     }
103     return buffer;
104 }
```

Here is the caller graph for this function:



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

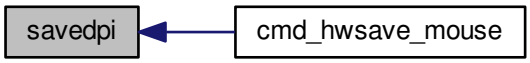
Definition at line 194 of file dpi.c.

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().

```
194                                                     {
195     // Send X/Y DPIs
196     for(int i = 0; i < DPI_COUNT; i++){
197         uchar data_pkt[MSG_SIZE] = { 0x07, 0x13, 0xd0, 1 };
198         data_pkt[2] |= i;
199         *(ushort*)(data_pkt + 5) = dpi->x[i];
200         *(ushort*)(data_pkt + 7) = dpi->y[i];
201         // Save the RGB value for this setting too
202         data_pkt[9] = light->r[LED_MOUSE + N_MOUSE_ZONES + i];
203         data_pkt[10] = light->g[LED_MOUSE + N_MOUSE_ZONES + i];
204         data_pkt[11] = light->b[LED_MOUSE + N_MOUSE_ZONES + i];
205         if(!usbsend(kb, data_pkt, 1))
206             return -1;
207     }
208
209     // Send settings
210     uchar data_pkt[4][MSG_SIZE] = {
211         { 0x07, 0x13, 0x05, 1, dpi->enabled },
212         { 0x07, 0x13, 0x02, 1, dpi->current },
213         { 0x07, 0x13, 0x03, 1, dpi->lift },
214         { 0x07, 0x13, 0x04, 1, dpi->snap, 0x05 }
215     };
216     if(!usbsend(kb, data_pkt[0], 4))
217         return -2;
218     // Finished
219     return 0;
220 }
```

Here is the caller graph for this function:



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

Definition at line 106 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.

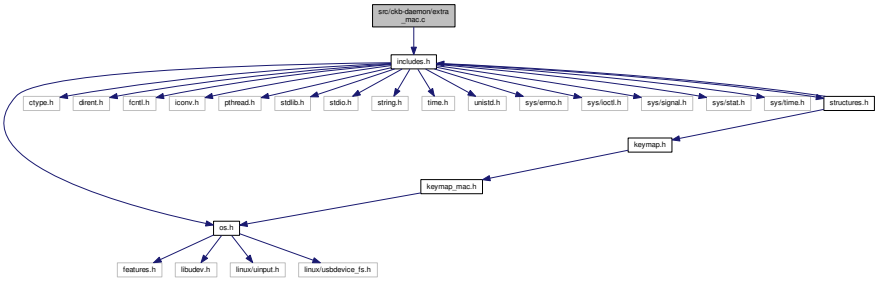
```
106                                         {
107     if(!kb->active)
108         return 0;
109     dpiset* lastdpi = &kb->profile->lastdpi;
110     dpiset* newdpi = &kb->profile->currentmode->dpi;
111     // Don't do anything if the settings haven't changed
112     if(!force && !lastdpi->forceupdate && !newdpi->forceupdate
113             && !memcmp(lastdpi, newdpi, sizeof(dpiset)))
114         return 0;
115     lastdpi->forceupdate = newdpi->forceupdate = 0;
116
117     if (newdpi->current != lastdpi->current) {
118         // Before we switch the current DPI stage, make sure the stage we are
119         // switching to is both enabled and configured to the correct DPI.
120
121         // Enable the stage if necessary.
122         if ((lastdpi->enabled & 1 << newdpi->current) == 0) {
123             uchar newenabled;
124             // If the new enabled flags contain both the current and previous
125             // stages, use it.
126             if (newdpi->enabled & 1 << newdpi->current &&
127                 newdpi->enabled & 1 << lastdpi->current) {
128                 newenabled = newdpi->enabled;
129             } else {
130                 // Otherwise just enable the new stage. We'll write the actual
131                 // requested flags after switching stages.
```

```
132                 newenabled = lastdpi->enabled | 1 << newdpi->current;
133             }
134             uchar data_pkt[MSG_SIZE] = { 0x07, 0x13, 0x05, 0, newenabled };
135             if(!usbsend(kb, data_pkt, 1))
136                 return -2;
137             // Cache the flags we wrote.
138             lastdpi->enabled = newenabled;
139         }
140         // Set the DPI for the new stage if necessary.
141         if (newdpi->x[newdpi->current] != lastdpi->x[newdpi->current] ||
142             newdpi->y[newdpi->current] != lastdpi->y[newdpi->current]) {
143             uchar data_pkt[MSG_SIZE] = { 0x07, 0x13, 0xd0, 0 };
144             data_pkt[2] |= newdpi->current;
145             *(ushort*)(data_pkt + 5) = newdpi->x[newdpi->current];
146             *(ushort*)(data_pkt + 7) = newdpi->y[newdpi->current];
147             if(!usbsend(kb, data_pkt, 1))
148                 return -1;
149             // Set these values in the cache so we don't rewrite them.
150             lastdpi->x[newdpi->current] = newdpi->x[newdpi->current];
151             lastdpi->y[newdpi->current] = newdpi->y[newdpi->current];
152         }
153         // Set current DPI stage.
154         uchar data_pkt[MSG_SIZE] = { 0x07, 0x13, 0x02, 0, newdpi->
    current };
155         if(!usbsend(kb, data_pkt, 1))
156             return -2;
157     }
158
159     // Send X/Y DPIs. We've changed to the new stage already so these can be set
160     // safely.
161     for(int i = 0; i < DPI_COUNT; i++){
162         if (newdpi->x[i] == lastdpi->x[i] && newdpi->y[i] == lastdpi->y[i])
163             continue;
164         uchar data_pkt[MSG_SIZE] = { 0x07, 0x13, 0xd0, 0 };
165         data_pkt[2] |= i;
166         *(ushort*)(data_pkt + 5) = newdpi->x[i];
167         *(ushort*)(data_pkt + 7) = newdpi->y[i];
168         if(!usbsend(kb, data_pkt, 1))
169             return -1;
170     }
171
172     // Send settings
173     if (newdpi->enabled != lastdpi->enabled) {
174         uchar data_pkt[MSG_SIZE] = { 0x07, 0x13, 0x05, 0, newdpi->
    enabled };
175         if(!usbsend(kb, data_pkt, 1))
176             return -2;
177     }
178     if (newdpi->lift != lastdpi->lift) {
179         uchar data_pkt[MSG_SIZE] = { 0x07, 0x13, 0x03, 0, newdpi->
    lift };
180         if(!usbsend(kb, data_pkt, 1))
181             return -2;
182     }
183     if (newdpi->snap != lastdpi->snap) {
184         uchar data_pkt[MSG_SIZE] = { 0x07, 0x13, 0x04, 0, newdpi->
    snap, 0x05 };
185         if(!usbsend(kb, data_pkt, 1))
186             return -2;
187     }
188
189     // Finished
190     memcpy(lastdpi, newdpi, sizeof(dpiset));
191     return 0;
192 }
```

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

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

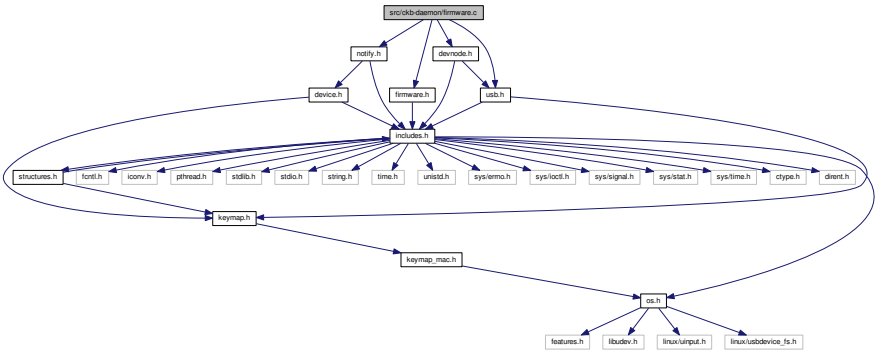
Include dependency graph for extra_mac.c:



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

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

Include dependency graph for firmware.c:



Macros

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

Functions

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

9.17.1 Macro Definition Documentation

9.17.1.1 #define FW_MAXSIZE (255 * 256)

Definition at line 51 of file firmware.c.

Referenced by fwupdate().

9.17.1.2 #define FW_NOFILE -1

Definition at line 7 of file firmware.c.

Referenced by cmd_fwupdate(), and fwupdate().

9.17.1.3 #define FW_OK 0

Definition at line 6 of file firmware.c.

Referenced by cmd_fwupdate(), and fwupdate().

9.17.1.4 #define FW_USBFAIL -3

Definition at line 9 of file firmware.c.

Referenced by cmd_fwupdate(), and fwupdate().

9.17.1.5 #define FW_WRONGDEV -2

Definition at line 8 of file firmware.c.

Referenced by cmd_fwupdate(), and fwupdate().

9.17.2 Function Documentation

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

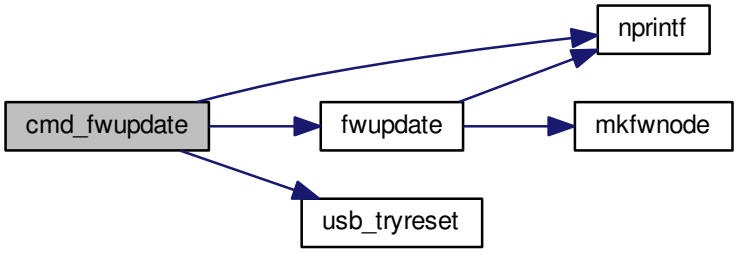
Definition at line 154 of file firmware.c.

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

```
154                                                                                                 {
155     (void)dummy1;
156     (void)dummy2;
157
158     if(!HAS_FEATURES(kb, FEAT_FWUPDATE))
159         return 0;
160     // Update the firmware
161     int ret = fwupdate(kb, path, nnumber);
162     while(ret == FW_USBFAIL){
163         // Try to reset the device if it fails
164         if(usb_tryreset(kb))
165             break;
166         ret = fwupdate(kb, path, nnumber);
167     }
168     switch(ret){
169     case FW_OK:
170         nprintf(kb, nnumber, 0, "fwupdate %s ok\n", path);
171         break;
172     case FW_NOFILE:
173     case FW_WRONGDEV:
174         nprintf(kb, nnumber, 0, "fwupdate %s invalid\n", path);
175         break;
176     case FW_USBFAIL:
177         nprintf(kb, nnumber, 0, "fwupdate %s fail\n", path);
```

```
178         return -1;
179     }
180     return 0;
181 }
```

Here is the call graph for this function:



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

Definition at line 55 of file firmware.c.

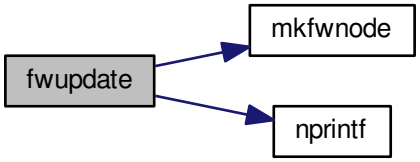
References ckb_err, ckb_info, FW_MAXSIZE, FW_NOFILE, FW_OK, FW_USBFAIL, FW_WRONGDEV, usbdevice::fwversion, mkfwnode(), MSG_SIZE, nprintf(), usbdevice::product, usbdevice::usbdelay, usbsend, and usbdevice::vendor.

Referenced by cmd_fwupdate().

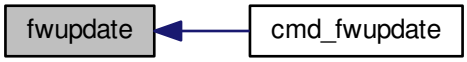
```
55                                                  {
56     // Read the firmware from the given path
57     char* fwdata = calloc(1, FW_MAXSIZE + 256);
58     int fd = open(path, O_RDONLY);
59     if(fd == -1){
60         ckb_err("Failed to open firmware file %s: %s\n", path, strerror(errno));
61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

```
91          { 0x7f, 0x03, 0x3c, 0 },
92          { 0x7f, 0x04, 0x3c, 0 },
93          { 0x7f, 0x05, 0x10, 0 }
94      };
95      int output = 0, last = 0;
96      int index = 0;
97      while(output < length){
98          int npackets = 1;
99          // Packet 1: data position
100         data_pkt[1][6] = index++;
101         while(output < length){
102             npackets++;
103             if(npackets != 6){
104                 // Packets 2-5: 60 bytes of data
105                 memcpy(data_pkt[npackets] + 4, fwdata + output, 60);
106                 last = output;
107                 output += 60;
108             } else {
109                 // Packet 6: 16 bytes
110                 memcpy(data_pkt[npackets] + 4, fwdata + output, 16);
111                 last = output;
112                 output += 16;
113                 break;
114             }
115         }
116         if(index == 1){
117             if(!usbsend(kb, data_pkt[0], 1)){
118                 ckb_err("Firmware update failed\n");
119                 return FW_USBFAIL;
120             }
121             // The above packet can take a lot longer to process, so wait for a while
122             sleep(3);
123             if(!usbsend(kb, data_pkt[2], npackets - 1)){
124                 ckb_err("Firmware update failed\n");
125                 return FW_USBFAIL;
126             }
127         } else {
128             // If the output ends here, set the length byte appropriately
129             if(output >= length)
130                 data_pkt[npackets][2] = length - last;
131             if(!usbsend(kb, data_pkt[1], npackets)){
132                 ckb_err("Firmware update failed\n");
133                 return FW_USBFAIL;
134             }
135         }
136         nprintf(kb, nnumber, 0, "fwupdate %s %d/%d\n", path, output, (int)length);
137     }
138     // Send the final pair of messages
139     uchar data_pkt2[2][MSG_SIZE] = {
140         { 0x07, 0x0d, 0xf0, 0x00, 0x00, 0x00, index },
141         { 0x07, 0x02, 0xf0, 0 }
142     };
143     if(!usbsend(kb, data_pkt2[0], 2)){
144         ckb_err("Firmware update failed\n");
145         return FW_USBFAIL;
146     }
147     // Updated successfully
148     kb->fwversion = version;
149     mkfwnode(kb);
150     ckb_info("Firmware update complete\n");
151     return FW_OK;
152 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



9.17.2.3 int getfwversion (usbdevice * *kb*)

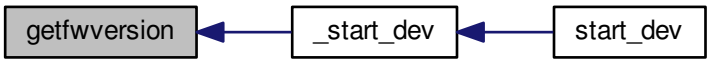
Definition at line 11 of file firmware.c.

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

Referenced by _start_dev().

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

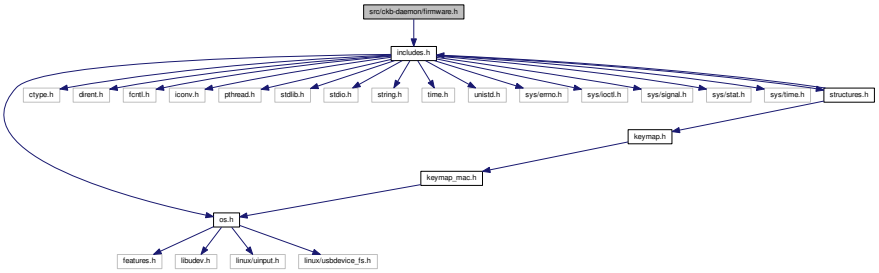
Here is the caller graph for this function:



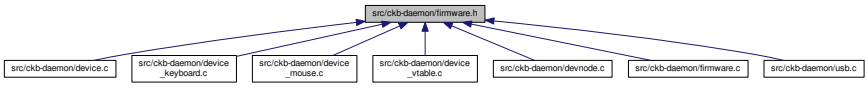
9.18 src/ckb-daemon/firmware.h File Reference

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

Include dependency graph for firmware.h:



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



Functions

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

9.18.1 Function Documentation

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

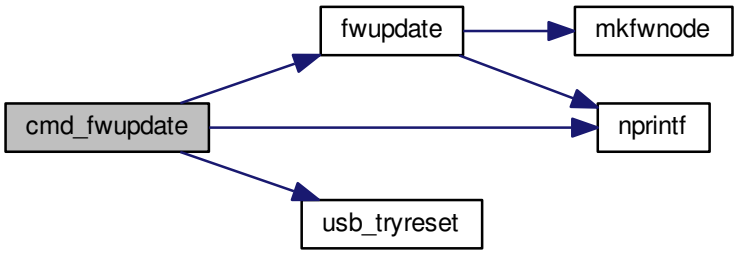
Definition at line 154 of file firmware.c.

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

```
154                                                                                 {
155     (void)dummy1;
156     (void)dummy2;
```

```
157
158     if(!HAS_FEATURES(kb, FEAT_FWUPDATE))
159         return 0;
160     // Update the firmware
161     int ret = fwupdate(kb, path, nnumber);
162     while(ret == FW_USBFAIL){
163         // Try to reset the device if it fails
164         if(usb_tryreset(kb))
165             break;
166         ret = fwupdate(kb, path, nnumber);
167     }
168     switch(ret){
169     case FW_OK:
170         nprintf(kb, nnumber, 0, "fwupdate %s ok\n", path);
171         break;
172     case FW_NOFILE:
173     case FW_WRONGDEV:
174         nprintf(kb, nnumber, 0, "fwupdate %s invalid\n", path);
175         break;
176     case FW_USBFAIL:
177         nprintf(kb, nnumber, 0, "fwupdate %s fail\n", path);
178         return -1;
179     }
180     return 0;
181 }
```

Here is the call graph for this function:



9.18.1.2 int getfwversion (usbdevice * *kb*)

Definition at line 11 of file firmware.c.

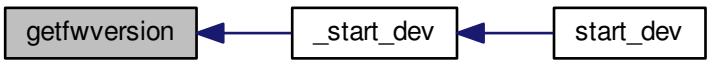
References ckb_err, ckb_warn, FEAT_POLLRATE, usbdevice::features, usbdevice::fwversion, MSG_SIZE, usbdevice::pollrate, usbdevice::product, usbrecv, and usbdevice::vendor.

Referenced by _start_dev().

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

```
29          poll = -1;
30          kb->features &= ~FEAT_POLLRATE;
31      }
32      // Print a warning if the message didn't match the expected data
33      if(vendor != kb->vendor)
34          ckb_warn("Got vendor ID %04x (expected %04x)\n", vendor, kb->
    vendor);
35      if(product != kb->product)
36          ckb_warn("Got product ID %04x (expected %04x)\n", product, kb->
    product);
37      // Set firmware version and poll rate
38      if(version == 0 || bootloader == 0){
39          // Needs firmware update
40          kb->fwversion = 0;
41          kb->pollrate = -1;
42      } else {
43          if(version != kb->fwversion && kb->fwversion != 0)
44              ckb_warn("Got firmware version %04x (expected %04x)\n", version, kb->
    fwversion);
45          kb->fwversion = version;
46          kb->pollrate = poll;
47      }
48      return 0;
49  }
```

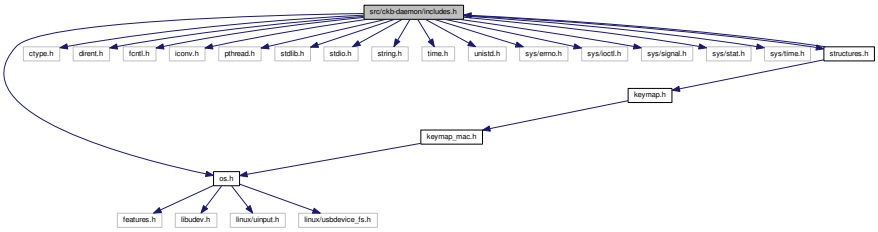
Here is the caller graph for this function:



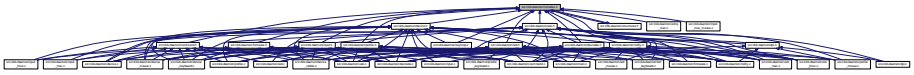
9.19 src/ckb-daemon/includes.h File Reference

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

Include dependency graph for includes.h:



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



Macros

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

Typedefs

- typedef unsigned char uchar
- typedef unsigned short ushort

Functions

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

9.19.1 Macro Definition Documentation

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

Definition at line 40 of file includes.h.

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

Definition at line 49 of file includes.h.

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

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

Definition at line 48 of file includes.h.

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

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

Definition at line 47 of file includes.h.

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

Definition at line 46 of file includes.h.

Referenced by usbmain().

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

Definition at line 45 of file includes.h.

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

Definition at line 44 of file includes.h.

Referenced by main().

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

Definition at line 55 of file includes.h.

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

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

Definition at line 54 of file includes.h.

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

Definition at line 53 of file includes.h.

Referenced by main().

9.19.1.11 #define ckb_s_err stdout

Definition at line 36 of file includes.h.

9.19.1.12 #define ckb_s_out stdout

Definition at line 35 of file includes.h.

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

Definition at line 52 of file includes.h.

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

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

Definition at line 51 of file includes.h.

Referenced by os_usbrecv(), and os_usbsend().

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

Definition at line 50 of file includes.h.

Referenced by main().

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

Definition at line 27 of file includes.h.

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

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

Definition at line 60 of file includes.h.

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

Definition at line 61 of file includes.h.

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

Definition at line 59 of file includes.h.

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

Definition at line 63 of file includes.h.

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

Definition at line 62 of file includes.h.

9.19.2 Typedef Documentation

9.19.2.1 typedef unsigned char uchar

Definition at line 24 of file includes.h.

9.19.2.2 typedef unsigned short ushort

Definition at line 25 of file includes.h.

9.19.3 Function Documentation

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

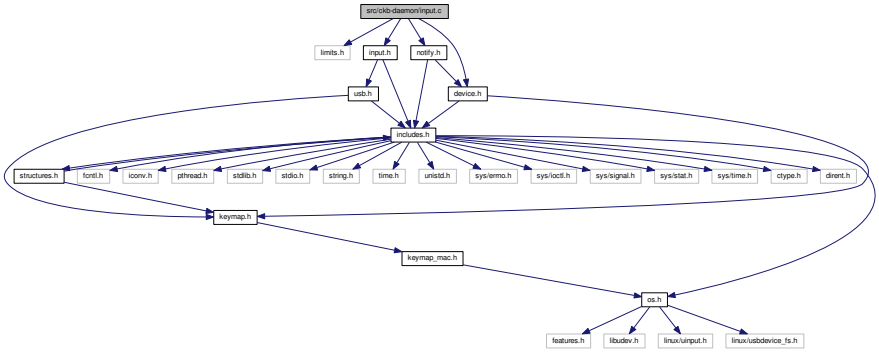
Definition at line 19 of file main.c.

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

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

```
#include <limits.h>
#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) && !IS_K63(kb)))

Functions

- int macromask (const uchar *key1, const uchar *key2)
- static void macro_pt_enqueue ()

  macro_pt_enqueue Save the new thread in the single linked list (FIFO).
- static pthread_t macro_pt_dequeue ()

  macro_pt_dequeue gets the first thread id of the list and returns the thread_id stored in it.
- static pthread_t macro_pt_first ()

  macro_pt_first returns the first pthread_id but does not remove the first entry.
- static void * play_macro (void *param)

  play_macro is the code for all threads started to play a macro.
- static void inputupdate_keys (usbdevice *kb)

  inputupdate_keys Handle input from Keyboard or mouse; start Macrof if detected.
- 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)

Variables

- static ptlist_t * pt_head = 0

  pt_head is the head pointer for the single linked thread list managed by macro_pt_en/dequeue().
- static ptlist_t * pt_tail = 0

  pt_tail is the tail pointer for the single linked thread list managed by macro_pt_en/dequeue().

9.20.1 Macro Definition Documentation

9.20.1.1 #define IS_WHEEL(*scan*, *kb*) (((scan) == KEY_VOLUMEUP || (scan) == KEY_VOLUMEDOWN || (scan) == BTN_WHEELUP || (scan) == BTN_WHEELDOWN) && (!IS_K65(kb) && !IS_K63(kb)))

Referenced by inputupdate_keys().

9.20.2 Function Documentation

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

Definition at line 353 of file input.c.

References keymacro::actioncount, keymacro::actions, usbmode::bind, keymacro::combo, macroaction::delay, 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().

```
353                                                                                 {
354     binding* bind = &mode->bind;
355     if(!keys && !assignment){
356         // Null strings = "macro clear" -> erase the whole thing
357         for(int i = 0; i < bind->macrocount; i++)
358             free(bind->macros[i].actions);
359         bind->macrocount = 0;
360         return;
361     }
362     if(bind->macrocount >= MACRO_MAX)
363         return;
364     // Create a key macro
365     keymacro macro;
366     memset(&macro, 0, sizeof(macro));
367     // Scan the left side for key names, separated by +
368     int empty = 1;
369     int left = strlen(keys), right = strlen(assignment);
370     int position = 0, field = 0;
371     char keyname[24];
372     while(position < left && sscanf(keys + position, "%10[^+]%n", keyname, &field) == 1){
373         int keycode;
374         if((sscanf(keyname, "#%d", &keycode) && keycode >= 0 && keycode <
    N_KEYS_INPUT)
375                 || (sscanf(keyname, "#x%x", &keycode) && keycode >= 0 && keycode <
    N_KEYS_INPUT)){
376             // Set a key numerically
377             SET_KEYBIT(macro.combo, keycode);
378             empty = 0;
379         } else {
380             // Find this key in the keymap
381             for(unsigned i = 0; i < N_KEYS_INPUT; i++){
382                 if(keymap[i].name && !strcmp(keyname, keymap[i].name)){
383                     macro.combo[i / 8] |= 1 << (i % 8);
384                     empty = 0;
385                     break;
386                 }
387             }
388         }
389         if(keys[position += field] == '+')
390             position++;
391     }
392     if(empty)
393         return;
394     // Count the number of actions (comma separated)
395     int count = 1;
396     for(const char* c = assignment; *c != 0; c++){
397         if(*c == ',')
398             count++;
399     }
400     // Allocate a buffer for them
401     macro.actions = calloc(count, sizeof(macroaction));
402     macro.actioncount = 0;
403     // Scan the actions
404     position = 0;
405     field = 0;
406     // max action = old 11 chars plus 12 chars which is the max 32-bit int 4294967295 size
407     while(position < right && sscanf(assignment + position, "%23[^,]%n", keyname, &field) == 1){
408         if(!strcmp(keyname, "clear"))
409             break;
```

```
410
411         // Check for local key delay of the form '[+-]<key>=<delay>'
412         long int long_delay;    // scanned delay value, used to keep delay in range.
413         unsigned int delay = UINT_MAX; // computed delay value. UINT_MAX means use global delay value.
414         char real_keyname[12];  // temp to hold the left side (key) of the <key>=<delay>
415         int scan_matches = sscanf(keyname, "%11[^=]=%ld", real_keyname, &long_delay);
416         if (scan_matches == 2) {
417             if (0 <= long_delay && long_delay < UINT_MAX) {
418                 delay = (unsigned int)long_delay;
419                 strcpy(keyname, real_keyname); // keyname[24], real_keyname[12]
420             }
421         }
422
423         int down = (keyname[0] == '+');
424         if(down || keyname[0] == '-'){
425             int keycode;
426             if((sscanf(keyname + 1, "#%d", &keycode) && keycode >= 0 && keycode < N_KEYS_INPUT)
427                     || (sscanf(keyname + 1, "#x%x", &keycode) && keycode >= 0 && keycode <
    N_KEYS_INPUT)){
428                 // Set a key numerically
429                 macro.actions[macro.actioncount].scan =
    keymap[keycode].scan;
430                 macro.actions[macro.actioncount].down = down;
431                 macro.actions[macro.actioncount].delay = delay;
432                 macro.actioncount++;
433             } else {
434                 // Find this key in the keymap
435                 for(unsigned i = 0; i < N_KEYS_INPUT; i++){
436                     if(keymap[i].name && !strcmp(keyname + 1, keymap[i].name)){
437                         macro.actions[macro.actioncount].scan =
    keymap[i].scan;
438                         macro.actions[macro.actioncount].down = down;
439                         macro.actions[macro.actioncount].delay = delay;
440                         macro.actioncount++;
441                         break;
442                     }
443                 }
444             }
445         }
446         if(assignment[position += field] == ',')
447             position++;
448     }
449
450     // See if there's already a macro with this trigger
451     keymacro* macros = bind->macros;
452     for(int i = 0; i < bind->macrocount; i++){
453         if(!memcmp(macros[i].combo, macro.combo, N_KEYBYTES_INPUT)){
454             free(macros[i].actions);
455             // If the new macro has no actions, erase the existing one
456             if(!macro.actioncount){
457                 for(int j = i + 1; j < bind->macrocount; j++)
458                     memcpy(macros + j - 1, macros + j, sizeof(keymacro));
459                 bind->macrocount--;
460             } else
461                 // If there are actions, replace the existing with the new
462                 memcpy(macros + i, &macro, sizeof(keymacro));
463             return;
464         }
465     }
466
467     // Add the macro to the device settings if not empty
468     if(macro.actioncount < 1)
469         return;
470     memcpy(bind->macros + (bind->macrocount++), &macro, sizeof(
    keymacro));
471     if(bind->macrocount >= bind->macrocap)
472         bind->macros = realloc(bind->macros, (bind->macrocap += 16) * sizeof(
    keymacro));
473 }
```

Here is the caller graph for this function:

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

Definition at line 307 of file input.c.

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

```
307                                                                                 {
308     (void)dummy;
309
310     if(keyindex >= N_KEYS_INPUT)
311         return;
312     // Find the key to bind to
313     int tocode = 0;
314     if(sscanf(to, "#x%ux", &tocode) != 1 && sscanf(to, "#%u", &tocode) == 1 && tocode <
    N_KEYS_INPUT){
315         pthread_mutex_lock(imutex(kb));
316         mode->bind.base[keyindex] = tocode;
317         pthread_mutex_unlock(imutex(kb));
318         return;
319     }
320     // If not numeric, look it up
321     for(int i = 0; i < N_KEYS_INPUT; i++){
322         if(keymap[i].name && !strcmp(to, keymap[i].name)){
323             pthread_mutex_lock(imutex(kb));
324             mode->bind.base[keyindex] = keymap[i].scan;
325             pthread_mutex_unlock(imutex(kb));
326             return;
327         }
328     }
329 }
```

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

Definition at line 475 of file input.c.

References _cmd_macro(), and imutex.

```
475
         {
476     (void)notifynumber;
477
478     pthread_mutex_lock(imutex(kb));
479     _cmd_macro(mode, keys, assignment);
480     pthread_mutex_unlock(imutex(kb));
481 }
```

Here is the call graph for this function:

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

Definition at line 342 of file input.c.

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

```
342                                                                                 {
```

```
343     (void)dummy;
344     (void)to;
345
346     if(keyindex >= N_KEYS_INPUT)
347         return;
348     pthread_mutex_lock(imutex(kb));
349     mode->bind.base[keyindex] = keymap[keyindex].scan;
350     pthread_mutex_unlock(imutex(kb));
351 }
```

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

Definition at line 331 of file input.c.

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

```
331                                                                                 {
332     (void)dummy;
333     (void)to;
334
335     if(keyindex >= N_KEYS_INPUT)
336         return;
337     pthread_mutex_lock(imutex(kb));
338     mode->bind.base[keyindex] = KEY_UNBOUND;
339     pthread_mutex_unlock(imutex(kb));
340 }
```

9.20.2.6 void freebind (binding * *bind*)

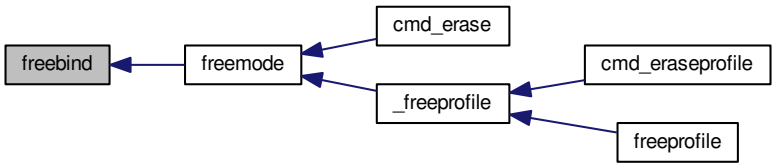
Definition at line 300 of file input.c.

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

Referenced by freemode().

```
300                             {
301     for(int i = 0; i < bind->macrocount; i++)
302         free(bind->macros[i].actions);
303     free(bind->macros);
304     memset(bind, 0, sizeof(*bind));
305 }
```

Here is the caller graph for this function:



9.20.2.7 void initbind (binding * *bind*)

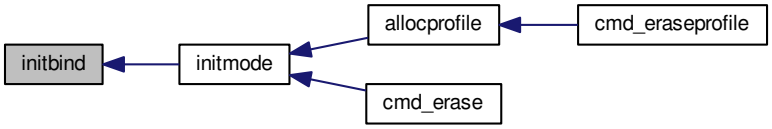
Definition at line 292 of file input.c.

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

Referenced by initmode().

```
292                             {
293     for(int i = 0; i < N_KEYS_INPUT; i++)
294         bind->base[i] = keymap[i].scan;
295     bind->macros = calloc(32, sizeof(keymacro));
296     bind->macrocap = 32;
297     bind->macrocount = 0;
298 }
```

Here is the caller graph for this function:



9.20.2.8 void inputupdate (usbdevice * *kb*)

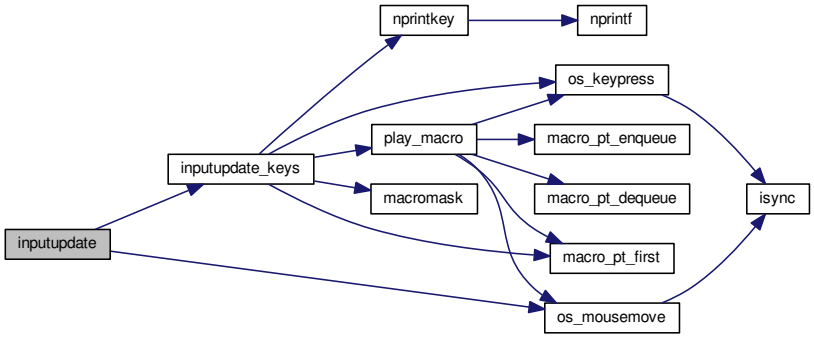
Definition at line 241 of file input.c.

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

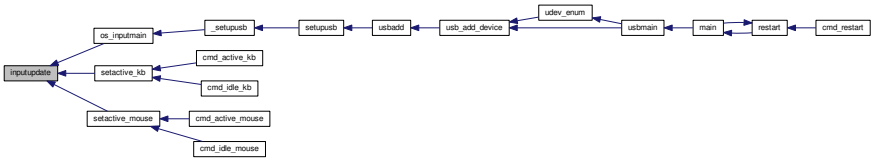
Referenced by os_inputmain(), setactive_kb(), and setactive_mouse().

```
241                                 {
242 #ifdef OS_LINUX
243     if((!kb->uinput_kb || !kb->uinput_mouse)
244 #else
245     if(!kb->event
246 #endif
247             || !kb->profile)
248         return;
249     // Process key/button input
250     inputupdate_keys(kb);
251     // Process mouse movement
252     usbinput* input = &kb->input;
253     if(input->rel_x != 0 || input->rel_y != 0){
254         os_mousemove(kb, input->rel_x, input->rel_y);
255         input->rel_x = input->rel_y = 0;
256     }
257     // Finish up
258     memcpy(input->prevkeys, input->keys, N_KEYBYTES_INPUT);
259 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



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

Parameters

kb	

Process all queued keypresses if no macro is running yet.

Todo If we want to get all keys typed while a macro is played, add the code for it here.

Definition at line 134 of file input.c.

References usbdevice::active, binding::base, usbmode::bind, keymacro::combo, usbprofile::currentmode, usbdevice::input, IS_MOD, IS_WHEEL, parameter::kb, keymap, usbinput::keys, parameter::macro, macro_pt_first(), binding::macrocount, macromask(), binding::macros, N_KEYBYTES_INPUT, N_KEYS_INPUT, usbmode::notify, nprintkey(), os_keypress(), OUTFIFO_MAX, play_macro(), usbinput::prevkeys, usbdevice::profile, key::scan, SCAN_SILENT, and keymacro::triggered.

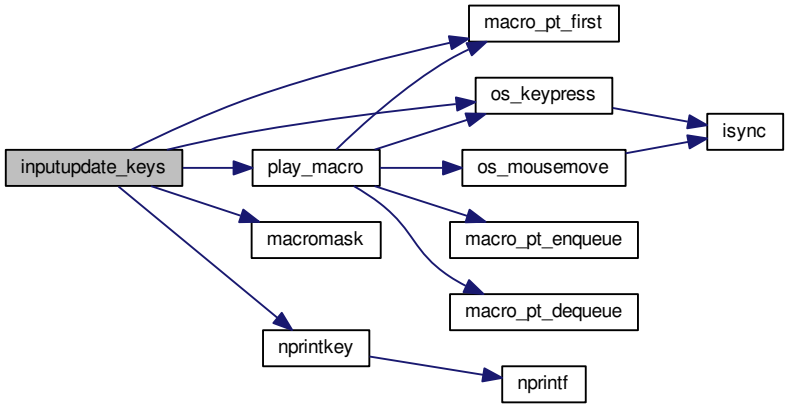
Referenced by inputupdate().

```
134                                         {
135     usbmode* mode = kb->profile->currentmode;
136     binding* bind = &mode->bind;
137     usbinput* input = &kb->input;
138 
139     // Don't do anything if the state hasn't changed
140     if(!memcmp(input->prevkeys, input->keys, N_KEYBYTES_INPUT))
141         return;
142     // Look for macros matching the current state
143     if (kb->active) {
144         for (int i = 0; i < bind->macrocount; i++) {
145             keymacro* macro = &bind->macros[i];
146             if (macromask(input->keys, macro->combo)) {
```

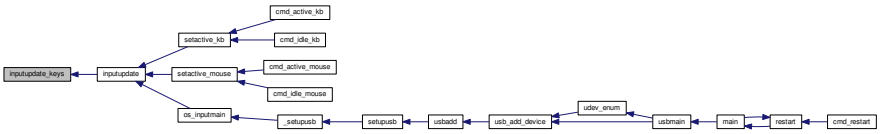
```
147                 if (!macro->triggered) {
148                     parameter_t* params = malloc(sizeof(parameter_t));
149                     if (params == 0) {
150                         perror("inputupdate_keys got no more mem:");
151                     } else {
152                         pthread_t thread = 0;
153                         params->kb = kb;
154                         params->macro = macro;
155                         int retval = pthread_create(&thread, 0, play_macro, (void*)params);
156                         if (retval) {
157                             perror("inputupdate_keys: Creating thread returned not null");
158                         } else {
159                             macro->triggered = 1;
160                         }
161                     }
162                 }
163             } else macro->triggered = 0;
164         }
165     }
166     // Make a list of keycodes to send. Rearrange them so that modifier keydowns always come first
167     // and modifier keyups always come last. This ensures that shortcut keys will register properly
168     // even if both keydown events happen at once.
169     // N_KEYS + 4 is used because the volume wheel generates keydowns and keyups at the same time
170     // (it's currently impossible to press all four at once, but safety first)
171     int events[N_KEYS_INPUT + 4];
172     int modcount = 0, keycount = 0, rmodcount = 0;
173     for(int byte = 0; byte < N_KEYBYTES_INPUT; byte++){
174         char oldb = input->prevkeys[byte], newb = input->keys[byte];
175         if(oldb == newb)
176             continue;
177         for(int bit = 0; bit < 8; bit++){
178             int keyindex = byte * 8 + bit;
179             if(keyindex >= N_KEYS_INPUT)
180                 break;
181             const key* map = keymap + keyindex;
182             int scancode = (kb->active) ? bind->base[keyindex] : map->
    scan;
183             char mask = 1 << bit;
184             char old = oldb & mask, new = newb & mask;
185             // If the key state changed, send it to the input device
186             if(old != new){
187                 // Don't echo a key press if there's no scancode associated
188                 if(!(scancode & SCAN_SILENT)){
189                     if(IS_MOD(scancode)){
190                         if(new){
191                             // Modifier down: Add to the end of modifier keys
192                             for(int i = keycount + rmodcount; i > 0; i--)
193                                 events[modcount + i] = events[modcount + i - 1];
194                             // Add 1 to the scancode because A is zero on OSX
195                             // Positive code = keydown, negative code = keyup
196                             events[modcount++] = scancode + 1;
197                         } else {
198                             // Modifier up: Add to the end of everything
199                             events[modcount + keycount + rmodcount++] = -(scancode + 1);
200                         }
201                     } else {
202                         // Regular keypress: add to the end of regular keys
203                         for(int i = rmodcount; i > 0; i--)
204                             events[modcount + keycount + i] = events[modcount + keycount + i - 1];
205                         events[modcount + keycount++] = new ? (scancode + 1) : -(scancode + 1);
206                         // The volume wheel and the mouse wheel don't generate keyups, so create them
    automatically
207 #define IS_WHEEL(scan, kb)  (((scan) == KEY_VOLUMEUP || (scan) == KEY_VOLUMEDOWN || (scan) == BTN_WHEELUP
    || (scan) == BTN_WHEELDOWN) && (!IS_K65(kb) && !IS_K63(kb)))
208                         if(new && IS_WHEEL(map->scan, kb)){
209                             for(int i = rmodcount; i > 0; i--)
210                                 events[modcount + keycount + i] = events[modcount + keycount + i - 1];
211                             events[modcount + keycount++] = -(scancode + 1);
212                             input->keys[byte] &= ~mask;
213                         }
214                     }
215                 }
216                 // Print notifications if desired
217                 if(kb->active){
218                     for(int notify = 0; notify < OUTFIFO_MAX; notify++){
219                         if(mode->notify[notify][byte] & mask){
220                             nprintkey(kb, notify, keyindex, new);
221                             // Wheels doesn't generate keyups
222                             if(new && IS_WHEEL(map->scan, kb))
223                                 nprintkey(kb, notify, keyindex, 0);
224                         }
225                     }
226                 }
227             }
228         }
229     }
232     if (!macro_pt_first()) {
```

```
233         int totalkeys = modcount + keycount + rmodcount;
234         for(int i = 0; i < totalkeys; i++){
235             int scancode = events[i];
236             os_keypress(kb, (scancode < 0 ? -scancode : scancode) - 1, scancode > 0);
237         }
238     }
239 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



9.20.2.10 static pthread_t macro_pt_dequeue ( ) [static]

Returns

the ptread_id of the first element. If list is empty, return 0.

Attention

Becuase multiple threads may use this function in parallel, save the critical section with a mutex.

< why are we called?

< Was last element in the list, so clear tail.

< save the return value before deleting element

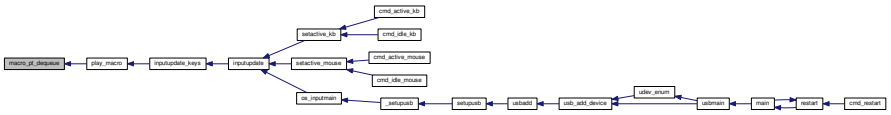
Definition at line 49 of file input.c.

References ckb_err, ptlist::next, pt_head, and ptlist::thread_id.

Referenced by play_macro().

```
49                                      {
50     pthread_t retval = 0;
51     ptlist_t* elem = 0;
52     if (pt_head == 0 && pt_tail == 0) {
53         ckb_err("macro_pt_dequeue: called on empty list.\n");
54         return 0;
55     }
56     elem = pt_head;
57     pt_head = pt_head->next;
58     if (pt_head == 0) pt_tail = 0;
59     retval = elem->thread_id;
60     free(elem);
61     return retval;
62 }
```

Here is the caller graph for this function:



9.20.2.11 static void macro_pt_enqueue ( ) [static]

Attention

Becuase multiple threads may use this function in parallel, save the critical section with a mutex.

< exit on critical situation;

Todo find a better exit strategy if no more mem available.

< The element knows its ID byself

< new list, first element

< existing list, append on last element (FIFO)

Definition at line 27 of file input.c.

References ptlist::next, and ptlist::thread_id.

Referenced by play_macro().

```
27                                      {
28     ptlist_t* new_elem = malloc(sizeof(ptlist_t));
29     if (!new_elem) {
30         perror("macro_pt_enqueue: ");
31         exit (-1);
32     }
33     new_elem->next = 0;
34     new_elem->thread_id = pthread_self();
35     if (pt_head == 0) {
36         pt_head = pt_tail = new_elem;
37     } else {
38         pt_tail->next = new_elem;
39         pt_tail = new_elem;
40     }
41 }
```

Here is the caller graph for this function:

9.20.2.12 static pthread_t macro_pt_first ( ) [static]

Returns

the pthread_id of the first element in the list or 0 if list is empty.

Attention

Becuase multiple threads may use this function in parallel, save the critical section with a mutex (avoid NIL-ptr)

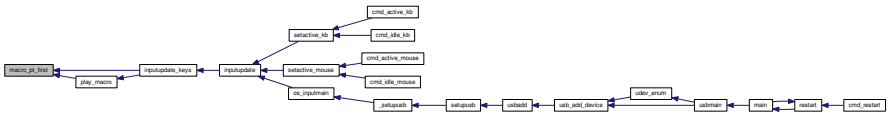
Definition at line 70 of file input.c.

References ptlist::thread_id.

Referenced by inputupdate_keys(), and play_macro().

```
70                                      {
71     return pt_head? pt_head->thread_id : 0;
72 }
```

Here is the caller graph for this function:



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

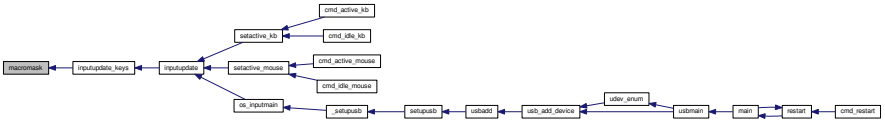
Definition at line 6 of file input.c.

References N_KEYBYTES_INPUT.

Referenced by inputupdate_keys().

```
6                                                       {
7     // Scan a macro against key input. Return 0 if any of them don't match
8     for(int i = 0; i < N_KEYBYTES_INPUT; i++){
9         // if((key1[i] & key2[i]) != key2[i])
10         if(key1[i] != key2[i])  // Changed to detect G-keys + modifiers
11             return 0;
12     }
13     return 1;
14 }
```

Here is the caller graph for this function:



9.20.2.14 static void* play_macro (void * param) [static]

Parameters

param	*parameter_t* to store Kb-ptr and macro-ptr (thread may get only one user-parameter)

Returns

0 on success, -1 else (no one is interested in it except the kernel...)

First have a look if we are the first and only macro-thread to run. If not, wait. So enqueue our thread first, so it is remembered for us and can be seen by all others.

< If the first thread in the list is not our, another one is running

< Give all new threads the chance to enter the block.

Send events for each keypress in the macro

< Synchonization between macro output and color information

< use this unlock / relock for enablling the parallel running colorization

< local delay set

< use default global delay

< use delays depending on macro length

< protect the linked list and the mvar

< Wake up all waiting threads

< for the linked list and the mvar

< Sync keyboard input/output and colorization

Definition at line 79 of file input.c.

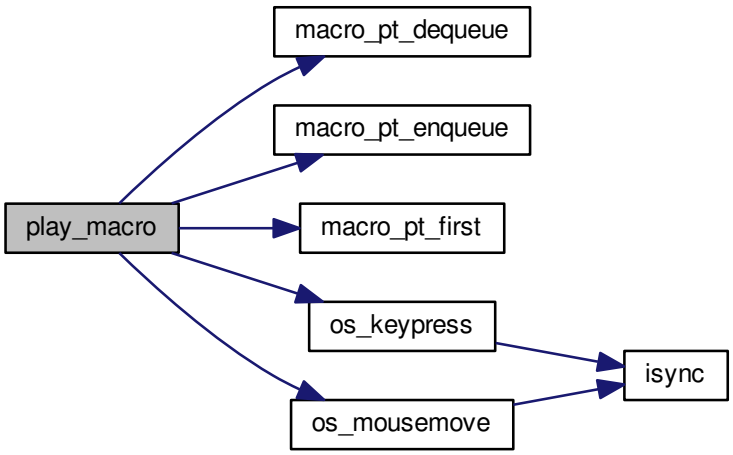
References keymacro::actioncount, keymacro::actions, macroaction::delay, usbdevice::delay, macroaction::down, parameter::kb, parameter::macro, macro_pt_dequeue(), macro_pt_enqueue(), macro_pt_first(), mmutex, mmutex2, mvar, os_keypress(), os_mousemove(), macroaction::rel_x, macroaction::rel_y, and macroaction::scan.

Referenced by inputupdate_keys().

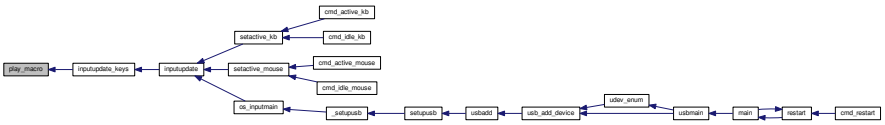
```
79                                      {
80     parameter_t* ptr = (parameter_t*) param;
81     usbdevice* kb = ptr->kb;
82     keymacro* macro = ptr->macro;
83 
86     pthread_mutex_lock(mmutex2(kb));
87     macro_pt_enqueue();
88     // ckb_info("Entering critical section with 0x%lx. Queue head is 0x%lx\n",  (unsigned long
       int)pthread_self(), (unsigned long int)macro_pt_first());
89     while (macro_pt_first() != pthread_self()) {
90         // ckb_info("Now waiting with 0x%lx because of 0x%lx\n", (unsigned long int)pthread_self(),
       (unsigned long int)macro_pt_first());
91         pthread_cond_wait(mvar(kb), mmutex2(kb));
92         // ckb_info("Waking up with 0x%lx\n", (unsigned long int)pthread_self());
93     }
94     pthread_mutex_unlock(mmutex2(kb));
95 
97     pthread_mutex_lock(mmutex(kb));
98     for (int a = 0; a < macro->actioncount; a++) {
99         macroaction* action = macro->actions + a;
100         if (action->rel_x != 0 || action->rel_y != 0)
101             os_mousemove(kb, action->rel_x, action->rel_y);
102         else {
103             os_keypress(kb, action->scan, action->down);
104             pthread_mutex_unlock(mmutex(kb));
105             if (action->delay != UINT_MAX && action->delay) {
106                 clock_nanosleep(CLOCK_MONOTONIC, 0, &(struct timespec) {.tv_nsec = action->
       delay * 1000}, NULL);
107             } else if (kb->delay != UINT_MAX && kb->delay) {
108                 clock_nanosleep(CLOCK_MONOTONIC, 0, &(struct timespec) {.tv_nsec = kb->
       delay * 1000}, NULL);
109             } else if (a < (macro->actioncount - 1)) {
110                 if (a > 200) {
111                     clock_nanosleep(CLOCK_MONOTONIC, 0, &(struct timespec) {.tv_nsec = action->
       delay * 100000}, NULL);
```

```
112                 } else if (a > 20) {
113                     clock_nanosleep(CLOCK_MONOTONIC, 0, &(struct timespec) {.tv_nsec = 30000}, NULL);
114                 }
115             }
116             pthread_mutex_lock(mmutex(kb));
117         }
118     }
119
120     pthread_mutex_lock(mmutex2(kb));
121     // ckb_info("Now leaving 0x%lx and waking up all others\n", (unsigned long int)pthread_self());
122     macro_pt_dequeue();
123     pthread_cond_broadcast(mvar(kb));
124     pthread_mutex_unlock(mmutex2(kb));
125
126     pthread_mutex_unlock(mmutex(kb));
127     return 0;
128 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



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

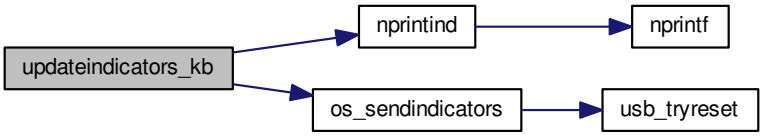
Definition at line 261 of file input.c.

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.

```
261                                                 {
262     // Read current hardware indicator state (set externally)
263     uchar old = kb->ileds, hw_old = kb->hw_ileds_old;
```

```
264     uchar new = kb->hw_ileds, hw_new = new;
265     // Update them if needed
266     if(kb->active){
267         usbmode* mode = kb->profile->currentmode;
268         new = (new & ~mode->ioff) | mode->ion;
269     }
270     kb->ileds = new;
271     kb->hw_ileds_old = hw_new;
272     if(old != new || force){
273         DELAY_SHORT(kb);
274         os_sendindicators(kb);
275     }
276     // Print notifications if desired
277     if(!kb->active)
278         return;
279     usbmode* mode = kb->profile->currentmode;
280     uchar indicators[] = { I_NUM, I_CAPS, I_SCROLL };
281     for(unsigned i = 0; i < sizeof(indicators) / sizeof(uchar); i++){
282         uchar mask = indicators[i];
283         if((hw_old & mask) == (hw_new & mask))
284             continue;
285         for(int notify = 0; notify < OUTFIFO_MAX; notify++){
286             if(mode->inotify[notify] & mask)
287                 nprintind(kb, notify, mask, hw_new & mask);
288         }
289     }
290 }
```

Here is the call graph for this function:



9.20.3 Variable Documentation

9.20.3.1 ptlist_t* pt_head = 0 [static]

Definition at line 18 of file input.c.

Referenced by macro_pt_dequeue().

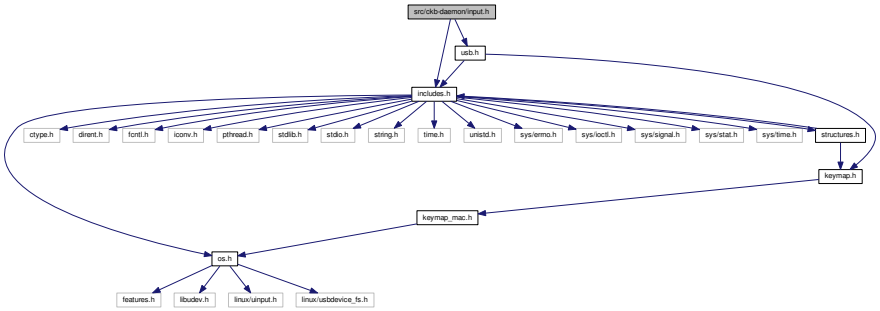
9.20.3.2 ptlist_t* pt_tail = 0 [static]

Definition at line 20 of file input.c.

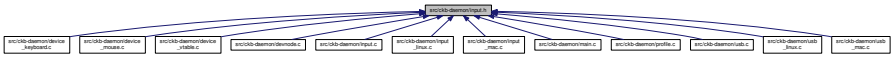
9.21 src/ckb-daemon/input.h File Reference

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

Include dependency graph for input.h:



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



Data Structures

- struct parameter

  struct parameter contains the values for a fresh started macro_play thread. parameter_t is the typedef for it. More...
- struct ptlist

  struct ptlist is one element in the single linked list to store macro_play threads waiting for their execution ptlist_t is the typedef for it. More...

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)

Typedefs

- typedef struct parameter parameter_t

  struct parameter contains the values for a fresh started macro_play thread. parameter_t is the typedef for it.
- typedef struct ptlist ptlist_t

  struct ptlist is one element in the single linked list to store macro_play threads waiting for their execution ptlist_t is the typedef for it.

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)

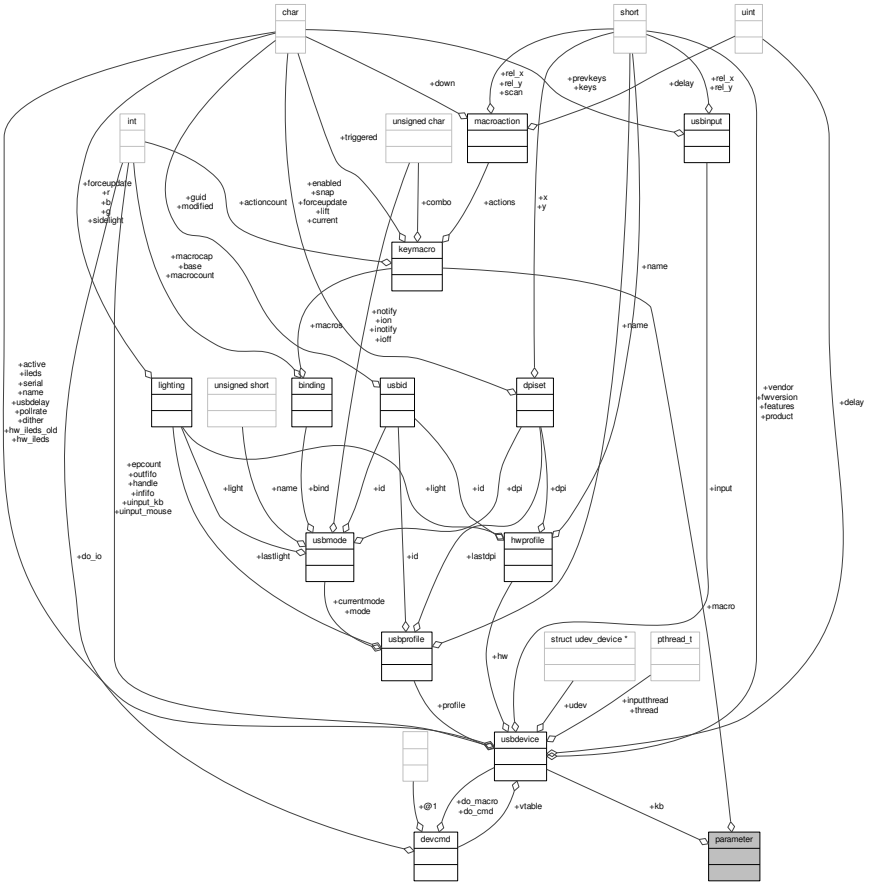
- void freebind (binding ∗bind)
- void cmd_bind (usbdevice ∗kb, usbmode ∗mode, int dummy, int keyindex, const char ∗to)
- void cmd_unbind (usbdevice ∗kb, usbmode ∗mode, int dummy, int keyindex, const char ∗ignored)
- void cmd_rebind (usbdevice ∗kb, usbmode ∗mode, int dummy, int keyindex, const char ∗ignored)
- 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)

9.21.1 Data Structure Documentation

9.21.1.1 struct parameter

Definition at line 54 of file input.h.

Collaboration diagram for parameter:



Data Fields

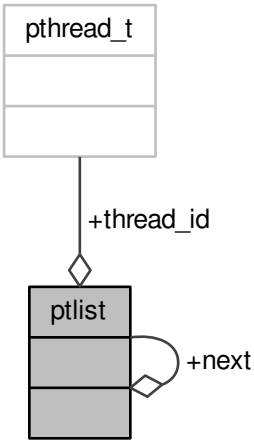
usbdevice ∗	kb	

keymacro *	macro	

9.21.1.2 struct ptlist

Definition at line 62 of file input.h.

Collaboration diagram for ptlist:



Data Fields

struct ptlist *	next	
pthread_t	thread_id	

9.21.2 Macro Definition Documentation

9.21.2.1 #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().

9.21.3 Typedef Documentation

9.21.3.1 typedef struct parameter parameter_t

9.21.3.2 typedef struct ptlist ptlist_t

9.21.4 Function Documentation

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

Definition at line 307 of file input.c.

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

```
307                                                                                 {
308     (void)dummy;
309
310     if(keyindex >= N_KEYS_INPUT)
311         return;
312     // Find the key to bind to
313     int tocode = 0;
314     if(sscanf(to, "#x%ux", &tocode) != 1 && sscanf(to, "#%u", &tocode) == 1 && tocode <
    N_KEYS_INPUT){
315         pthread_mutex_lock(imutex(kb));
316         mode->bind.base[keyindex] = tocode;
317         pthread_mutex_unlock(imutex(kb));
318         return;
319     }
320     // If not numeric, look it up
321     for(int i = 0; i < N_KEYS_INPUT; i++){
322         if(keymap[i].name && !strcmp(to, keymap[i].name)){
323             pthread_mutex_lock(imutex(kb));
324             mode->bind.base[keyindex] = keymap[i].scan;
325             pthread_mutex_unlock(imutex(kb));
326             return;
327         }
328     }
329 }
```

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

Definition at line 475 of file input.c.

References _cmd_macro(), and imutex.

```
475
         {
476     (void)notifynumber;
477
478     pthread_mutex_lock(imutex(kb));
479     _cmd_macro(mode, keys, assignment);
480     pthread_mutex_unlock(imutex(kb));
481 }
```

Here is the call graph for this function:

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

Definition at line 342 of file input.c.

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

```
342                                                                                 {
```

```
343     (void)dummy;
344     (void)to;
345
346     if(keyindex >= N_KEYS_INPUT)
347         return;
348     pthread_mutex_lock(imutex(kb));
349     mode->bind.base[keyindex] = keymap[keyindex].scan;
350     pthread_mutex_unlock(imutex(kb));
351 }
```

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

Definition at line 331 of file input.c.

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

```
331                                                                                 {
332     (void)dummy;
333     (void)to;
334
335     if(keyindex >= N_KEYS_INPUT)
336         return;
337     pthread_mutex_lock(imutex(kb));
338     mode->bind.base[keyindex] = KEY_UNBOUND;
339     pthread_mutex_unlock(imutex(kb));
340 }
```

9.21.4.5 void freebind (binding * *bind*)

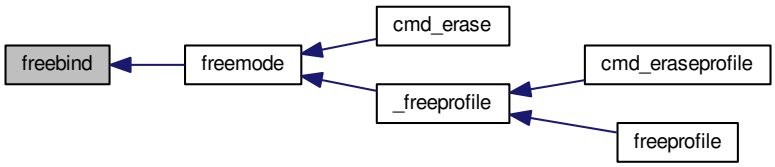
Definition at line 300 of file input.c.

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

Referenced by freemode().

```
300                             {
301     for(int i = 0; i < bind->macrocount; i++)
302         free(bind->macros[i].actions);
303     free(bind->macros);
304     memset(bind, 0, sizeof(*bind));
305 }
```

Here is the caller graph for this function:



9.21.4.6 void initbind (binding * *bind*)

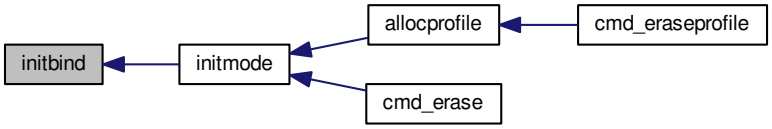
Definition at line 292 of file input.c.

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

Referenced by initmode().

```
292                             {
293     for(int i = 0; i < N_KEYS_INPUT; i++)
294         bind->base[i] = keymap[i].scan;
295     bind->macros = calloc(32, sizeof(keymacro));
296     bind->macrocap = 32;
297     bind->macrocount = 0;
298 }
```

Here is the caller graph for this function:



9.21.4.7 void inputupdate (usbdevice * *kb*)

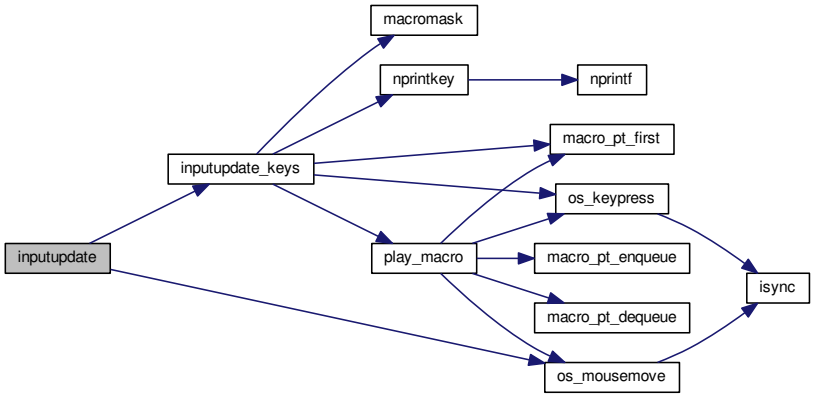
Definition at line 241 of file input.c.

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

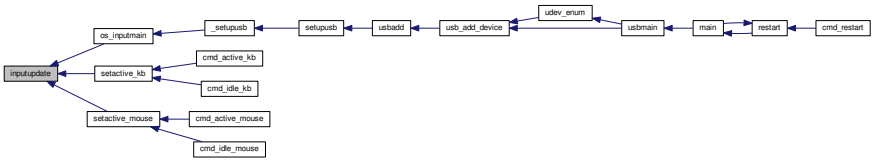
Referenced by os_inputmain(), setactive_kb(), and setactive_mouse().

```
241                                 {
242 #ifdef OS_LINUX
243     if((!kb->uinput_kb || !kb->uinput_mouse)
244 #else
245     if(!kb->event
246 #endif
247             || !kb->profile)
248         return;
249     // Process key/button input
250     inputupdate_keys(kb);
251     // Process mouse movement
252     usbinput* input = &kb->input;
253     if(input->rel_x != 0 || input->rel_y != 0){
254         os_mousemove(kb, input->rel_x, input->rel_y);
255         input->rel_x = input->rel_y = 0;
256     }
257     // Finish up
258     memcpy(input->prevkeys, input->keys, N_KEYBYTES_INPUT);
259 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



9.21.4.8 void os_inputclose (usbdevice * *kb*)

Definition at line 76 of file input_linux.c.

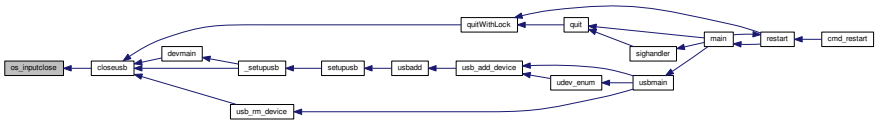
References ckb_warn, usbdevice::uinput_kb, and usbdevice::uinput_mouse.

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:



9.21.4.9 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().

```
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:

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

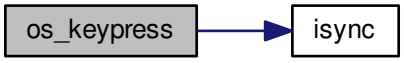
Definition at line 118 of file input_linux.c.

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

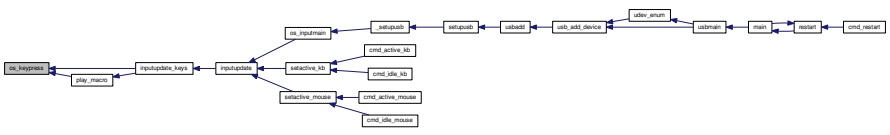
Referenced by inputupdate_keys(), and play_macro().

```
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:



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

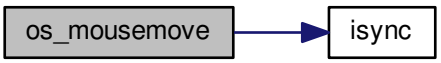
Definition at line 143 of file input_linux.c.

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

Referenced by inputupdate(), and play_macro().

```
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:

9.21.4.12 int os_setupindicators (usbdevice * kb)

Definition at line 189 of file input_linux.c.

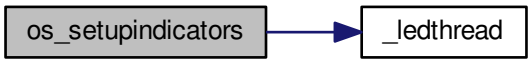
References _ledthread(), usbdevice::hw_ileds, usbdevice::hw_ileds_old, and usbdevice::ileds.

Referenced by _setupusb().

```
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)
```

```
196         return err;
197     pthread_detach(thread);
198     return 0;
199 }
```

Here is the call graph for this function:



Here is the caller graph for this function:

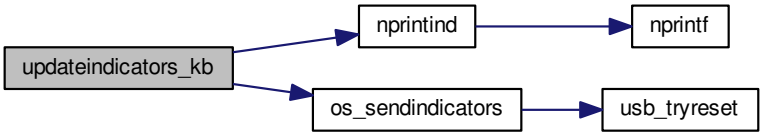
9.21.4.13 void updateindicators_kb (usbdevice * *kb,* int *force*)

Definition at line 261 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.

```
261                                                 {
262     // Read current hardware indicator state (set externally)
263     uchar old = kb->ileds, hw_old = kb->hw_ileds_old;
264     uchar new = kb->hw_ileds, hw_new = new;
265     // Update them if needed
266     if(kb->active){
267         usbmode* mode = kb->profile->currentmode;
268         new = (new & ~mode->ioff) | mode->ion;
269     }
270     kb->ileds = new;
271     kb->hw_ileds_old = hw_new;
272     if(old != new || force){
273         DELAY_SHORT(kb);
274         os_sendindicators(kb);
275     }
276     // Print notifications if desired
277     if(!kb->active)
278         return;
279     usbmode* mode = kb->profile->currentmode;
280     uchar indicators[] = { I_NUM, I_CAPS, I_SCROLL };
281     for(unsigned i = 0; i < sizeof(indicators) / sizeof(uchar); i++){
282         uchar mask = indicators[i];
283         if((hw_old & mask) == (hw_new & mask))
284             continue;
285         for(int notify = 0; notify < OUTFIFO_MAX; notify++){
286             if(mode->inotify[notify] & mask)
287                 nprintind(kb, notify, mask, hw_new & mask);
288         }
289     }
290 }
```

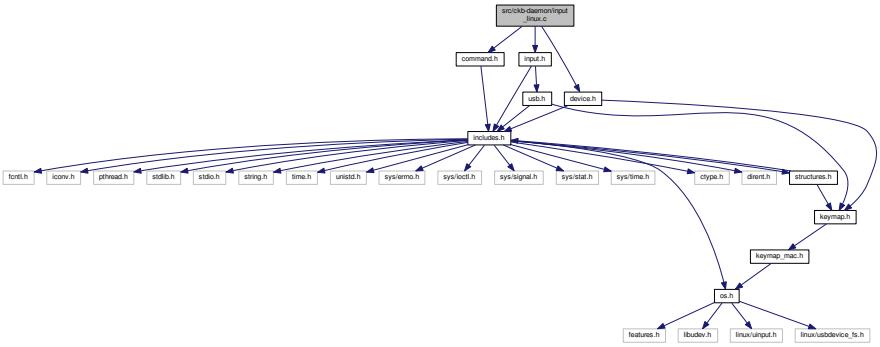
Here is the call graph for this function:



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

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

Include dependency graph for input_linux.c:



Functions

- int uinputopen (struct uinput_user_dev *indev, int mouse)
- int os_inputopen (usbdevice *kb)

  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)

9.22.1 Function Documentation

9.22.1.1 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:

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

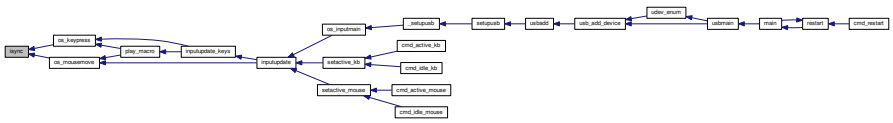
Definition at line 107 of file input_linux.c.

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:



9.22.1.3 void os_inputclose (usbdevice * kb)

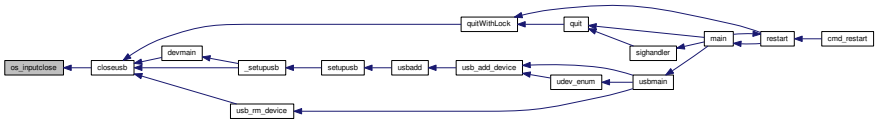
Definition at line 76 of file input_linux.c.

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:



9.22.1.4 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:

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

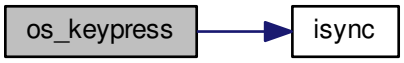
Definition at line 118 of file input_linux.c.

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

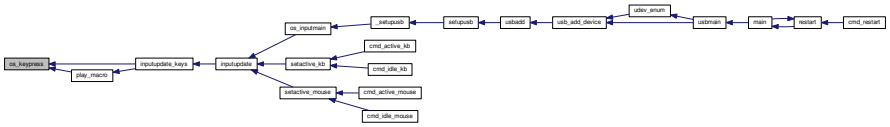
Referenced by inputupdate_keys(), and play_macro().

```
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:



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

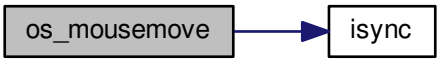
Definition at line 143 of file input_linux.c.

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

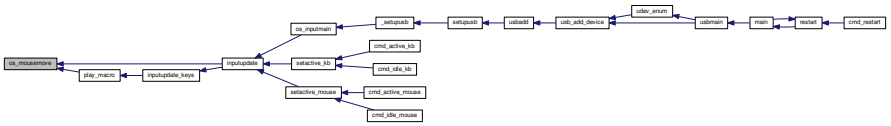
Referenced by inputupdate(), and play_macro().

```
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:



9.22.1.7 int os_setupindicators (usbdevice * *kb*)

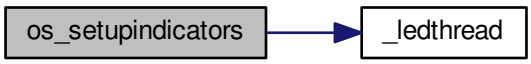
Definition at line 189 of file input_linux.c.

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)
196         return err;
197     pthread_detach(thread);
198     return 0;
199 }
```

Here is the call graph for this function:



Here is the caller graph for this function:

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

Definition at line 9 of file input_linux.c.

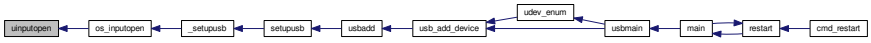
References ckb_err, and ckb_warn.

Referenced by os_inputopen().

```
9                                                       {
10     int fd = open("/dev/uinput", O_RDWR);
11     if(fd < 0){
12         // If that didn't work, try /dev/input/uinput instead
```

```
13          fd = open("/dev/input/uinput", O_RDWR);
14          if(fd < 0){
15              ckb_err("Failed to open uinput: %s\n", strerror(errno));
16              return 0;
17          }
18      }
19      // Enable all keys and mouse buttons
20      ioctl(fd, UI_SET_EVBIT, EV_KEY);
21      for(int i = 0; i < KEY_CNT; i++)
22          ioctl(fd, UI_SET_KEYBIT, i);
23      if(mouse){
24          // Enable mouse axes
25          ioctl(fd, UI_SET_EVBIT, EV_REL);
26          for(int i = 0; i < REL_CNT; i++)
27              ioctl(fd, UI_SET_RELBIT, i);
28      } else {
29          // Enable LEDs
30          ioctl(fd, UI_SET_EVBIT, EV_LED);
31          for(int i = 0; i < LED_CNT; i++)
32              ioctl(fd, UI_SET_LEDBIT, i);
33          // Eanble autorepeat
34          ioctl(fd, UI_SET_EVBIT, EV_REP);
35      }
36      // Enable sychronization
37      ioctl(fd, UI_SET_EVBIT, EV_SYN);
38      // Create the device
39      if(write(fd, indev, sizeof(*indev)) <= 0)
40          ckb_warn("uinput write failed: %s\n", strerror(errno));
41      if(ioctl(fd, UI_DEV_CREATE)){
42          ckb_err("Failed to create uinput device: %s\n", strerror(errno));
43          close(fd);
44          return 0;
45      }
46      return fd + 1;
47  }
```

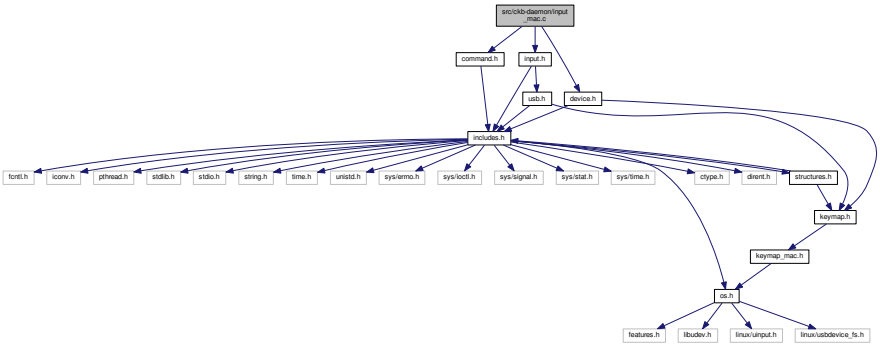
Here is the caller graph for this function:



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

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

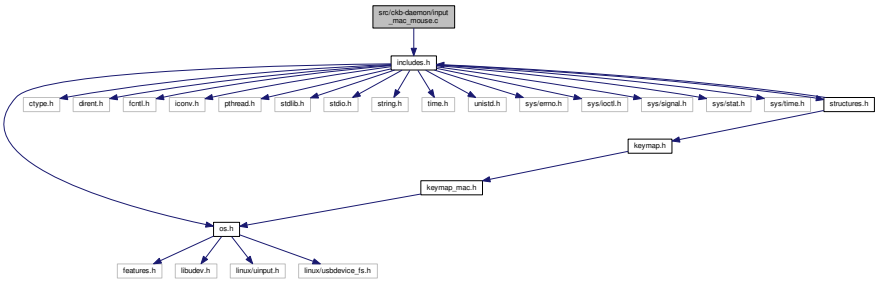
Include dependency graph for input_mac.c:



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

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

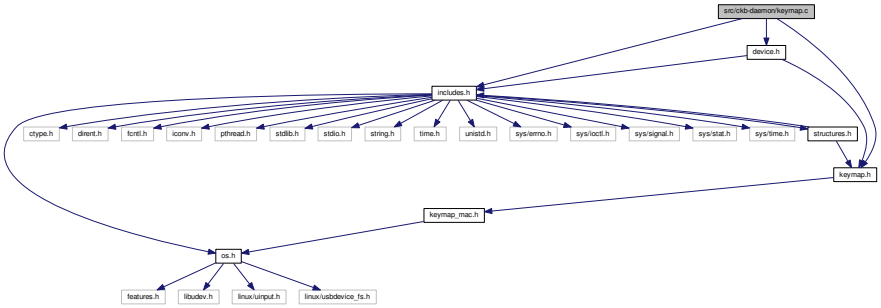
Include dependency graph for input_mac_mouse.c:



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

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

Include dependency graph for keymap.c:



Macros

- #define BUTTON_HID_COUNT 5

Functions

- void hid_kb_translate (unsigned char ∗kbinput, int endpoint, int length, const unsigned char ∗urbinput)
- void hid_mouse_translate (unsigned char ∗kbinput, short ∗xaxis, short ∗yaxis, int endpoint, int length, const unsigned char ∗urbinput, ushort fwversion)
- 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+22+12)+25)+12)]

9.25.1 Macro Definition Documentation

9.25.1.1 #define BUTTON_HID_COUNT 5

Definition at line 389 of file keymap.c.

Referenced by corsair_mousecopy(), and hid_mouse_translate().

9.25.2 Function Documentation

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

Definition at line 420 of file keymap.c.

References N_KEYBYTES_HW.

Referenced by os_inputmain().

```
420                                                                                                 {
421     if(endpoint == 2 || endpoint == -2){
422         if(urbinput[0] != 3)
423             return;
424         urbinput++;
425     }
426     memcpy(kbinput, urbinput, N_KEYBYTES_HW);
427 }
```

Here is the caller graph for this function:

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

Definition at line 429 of file keymap.c.

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

Referenced by os_inputmain().

```
429                                                                                                 {
430     if(endpoint == 2 || endpoint == -2){
431         if(urbinput[0] != 3)
432             return;
433         urbinput++;
434     }
435     for(int bit = BUTTON_HID_COUNT; bit < N_BUTTONS_HW; bit++){
436         int byte = bit / 8;
437         uchar test = 1 << (bit % 8);
438         if(urbinput[byte] & test)
439             SET_KEYBIT(kbinput, MOUSE_BUTTON_FIRST + bit);
440         else
441             CLEAR_KEYBIT(kbinput, MOUSE_BUTTON_FIRST + bit);
442     }
443 }
```

Here is the caller graph for this function:

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

Definition at line 246 of file keymap.c.

References ckb_warn, CLEAR_KEYBIT, and SET_KEYBIT.

Referenced by os_inputmain().

```
246                                                
```

```
321         } else if (urbinput[0] == 2)
322             ; // Type 2: media key (implicitly falls through)
323         else
324             break;  // No other known types
325         /* FALLTHRU */
326     case 2:
327         // EP 2 Non-RGB: media keys
328         CLEAR_KEYBIT(kbinput, 97);          // mute
329         CLEAR_KEYBIT(kbinput, 98);          // stop
330         CLEAR_KEYBIT(kbinput, 99);          // prev
331         CLEAR_KEYBIT(kbinput, 100);         // play
332         CLEAR_KEYBIT(kbinput, 101);         // next
333         CLEAR_KEYBIT(kbinput, 130);         // volup
334         CLEAR_KEYBIT(kbinput, 131);         // voldn
335         for(int i = 0; i < length; i++){
336             switch(urbinput[i]){
337             case 181:
338                 SET_KEYBIT(kbinput, 101);   // next
339                 break;
340             case 182:
341                 SET_KEYBIT(kbinput, 99);    // prev
342                 break;
343             case 183:
344                 SET_KEYBIT(kbinput, 98);    // stop
345                 break;
346             case 205:
347                 SET_KEYBIT(kbinput, 100);   // play
348                 break;
349             case 226:
350                 SET_KEYBIT(kbinput, 97);    // mute
351                 break;
352             case 233:
353                 SET_KEYBIT(kbinput, 130);   // volup
354                 break;
355             case 234:
356                 SET_KEYBIT(kbinput, 131);   // voldn
357                 break;
358             }
359         }
360         break;
361     case 3:
362         // EP 3 non-RGB: NKRO input
363         if(length != 15)
364             return;
365         for(int bit = 0; bit < 8; bit++){
366             if((urbinput[0] >> bit) & 1)
367                 SET_KEYBIT(kbinput, hid_codes[bit + 224]);
368             else
369                 CLEAR_KEYBIT(kbinput, hid_codes[bit + 224]);
370         }
371         for(int byte = 0; byte < 14; byte++){
372             char input = urbinput[byte + 1];
373             for(int bit = 0; bit < 8; bit++){
374                 int keybit = byte * 8 + bit;
375                 int scan = hid_codes[keybit];
376                 if((input >> bit) & 1){
377                     if(scan >= 0)
378                         SET_KEYBIT(kbinput, hid_codes[keybit]);
379                     else
380                         ckb_warn("Got unknown key press %d on EP 3\n", keybit);
381                 } else if(scan >= 0)
382                     CLEAR_KEYBIT(kbinput, hid_codes[keybit]);
383             }
384         }
385         break;
386     }
387 }
```

Here is the caller graph for this function:

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

Definition at line 391 of file keymap.c.

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

Referenced by os_inputmain().

```
391
                                                {
392     //The HID Input Endpoint on FWv3 is 64 bytes, so we can't check for length.
393     if((endpoint != 2 && endpoint != -2) || (fwversion < 0x300 && length < 10))
394         return;
395     // EP 2: mouse input
396     if(urbinput[0] != 1)
397         return;
398     // Byte 1 = mouse buttons (bitfield)
399     for(int bit = 0; bit < BUTTON_HID_COUNT; bit++){
400         if(urbinput[1] & (1 << bit))
401             SET_KEYBIT(kbinput, MOUSE_BUTTON_FIRST + bit);
402         else
403             CLEAR_KEYBIT(kbinput, MOUSE_BUTTON_FIRST + bit);
404     }
405     // Bytes 5 - 8: movement
406     *xaxis += *(short*)(urbinput + 5);
407     *yaxis += *(short*)(urbinput + 7);
408     // Byte 9: wheel
409     char wheel = urbinput[9];
410     if(wheel > 0)
411         SET_KEYBIT(kbinput, MOUSE_EXTRA_FIRST);         // wheelup
412     else
413         CLEAR_KEYBIT(kbinput, MOUSE_EXTRA_FIRST);
414     if(wheel < 0)
415         SET_KEYBIT(kbinput, MOUSE_EXTRA_FIRST + 1);     // wheeldn
416     else
417         CLEAR_KEYBIT(kbinput, MOUSE_EXTRA_FIRST + 1);
418 }
```

Here is the caller graph for this function:

9.25.3 Variable Documentation

9.25.3.1 const key keymap[(((152+22+12)+25)+12)]

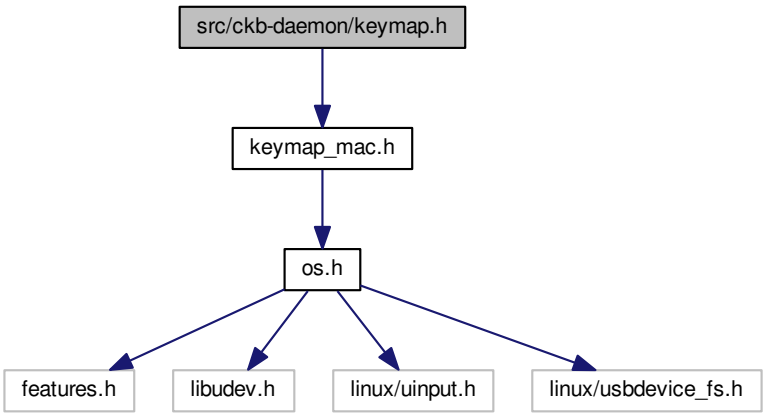
Definition at line 5 of file keymap.c.

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

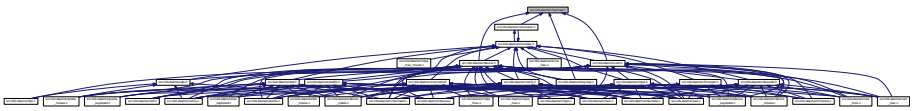
9.26 src/ckb-daemon/keymap.h File Reference

```
#include "keymap_mac.h"
```

Include dependency graph for keymap.h:



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



Data Structures

- struct key

Macros

- #define KEY_NONE -1
- #define KEY_CORSAIR -2
- #define KEY_UNBOUND -3
- #define BTN_WHEELUP 0x1f01
- #define BTN_WHEELDOWN 0x1f02
- #define KEY_BACKSLASH_ISO KEY_BACKSLASH
- #define N_KEYS_HW 152
- #define N_KEYBYTES_HW ((N_KEYS_HW + 7) / 8)
- #define N_KEY_ZONES 22
- #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 6
- #define N_MOUSE_ZONES_EXTENDED 12

- #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, ushort fwversion)
- 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

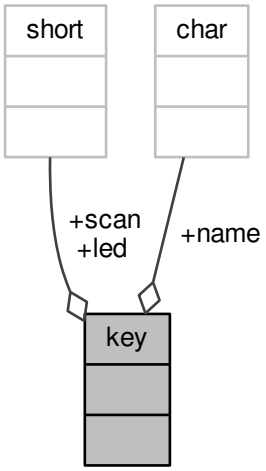
- const key keymap [(((152+22+12)+25)+12)]

9.26.1 Data Structure Documentation

9.26.1.1 struct key

Definition at line 49 of file keymap.h.

Collaboration diagram for key:



Data Fields

short	led	
const char *	name	
short	scan	

9.26.2 Macro Definition Documentation

9.26.2.1 #define BTN_WHEELDOWN 0x1f02

Definition at line 13 of file keymap.h.

Referenced by os_keypress().

9.26.2.2 #define BTN_WHEELUP 0x1f01

Definition at line 12 of file keymap.h.

Referenced by os_keypress().

9.26.2.3 #define KEY_BACKSLASH_ISO KEY_BACKSLASH

Definition at line 20 of file keymap.h.

9.26.2.4 #define KEY_CORSAIR -2

Definition at line 8 of file keymap.h.

9.26.2.5 #define KEY_NONE -1

Definition at line 7 of file keymap.h.

9.26.2.6 #define KEY_UNBOUND -3

Definition at line 9 of file keymap.h.

Referenced by cmd_unbind().

9.26.2.7 #define LED_DPI (LED_MOUSE + 2)

Definition at line 43 of file keymap.h.

Referenced by loadrgb_mouse(), and savergb_mouse().

9.26.2.8 #define LED_MOUSE N_KEYS_HW

Definition at line 39 of file keymap.h.

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

9.26.2.9 #define MOUSE_BUTTON_FIRST (N_KEYS_HW + N_KEY_ZONES + N_KEYS_EXTRA)

Definition at line 33 of file keymap.h.

Referenced by corsair_mousecopy(), and hid_mouse_translate().

9.26.2.10 #define MOUSE_EXTRA_FIRST (MOUSE_BUTTON_FIRST + N_BUTTONS_HW)

Definition at line 34 of file keymap.h.

Referenced by hid_mouse_translate().

9.26.2.11 #define N_BUTTONS_EXTENDED 25

Definition at line 32 of file keymap.h.

9.26.2.12 #define N_BUTTONS_HW 20

Definition at line 31 of file keymap.h.

Referenced by corsair_mousecopy().

9.26.2.13 #define N_KEY_ZONES 22

Definition at line 27 of file keymap.h.

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

Definition at line 46 of file keymap.h.

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

Definition at line 25 of file keymap.h.

Referenced by corsair_kbcopy().

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

Definition at line 37 of file keymap.h.

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

9.26.2.17 #define N_KEYS_EXTENDED (N_KEYS_INPUT + N_MOUSE_ZONES_EXTENDED)

Definition at line 45 of file keymap.h.

Referenced by printrgb(), and readcmd().

9.26.2.18 #define N_KEYS_EXTRA 12

Definition at line 29 of file keymap.h.

9.26.2.19 #define N_KEYS_HW 152

Definition at line 24 of file keymap.h.

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

9.26.2.20 #define N_KEYS_INPUT (MOUSE_BUTTON_FIRST + N_BUTTONS_EXTENDED)

Definition at line 36 of file keymap.h.

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

9.26.2.21 #define N_MOUSE_ZONES 6

Definition at line 40 of file keymap.h.

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

9.26.2.22 #define N_MOUSE_ZONES_EXTENDED 12

Definition at line 41 of file keymap.h.

9.26.2.23 #define SCAN_KBD 0

Definition at line 57 of file keymap.h.

9.26.2.24 #define SCAN_MOUSE 0x1000

Definition at line 58 of file keymap.h.

Referenced by os_keypress().

9.26.2.25 #define SCAN_SILENT 0x8000

Definition at line 56 of file keymap.h.

Referenced by inputupdate_keys().

9.26.3 Function Documentation

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

Definition at line 420 of file keymap.c.

References N_KEYBYTES_HW.

Referenced by os_inputmain().

```
420                                                                                 {
421     if(endpoint == 2 || endpoint == -2){
422         if(urbinput[0] != 3)
423             return;
424         urbinput++;
425     }
426     memcpy(kbinput, urbinput, N_KEYBYTES_HW);
427 }
```

Here is the caller graph for this function:

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

Definition at line 429 of file keymap.c.

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

Referenced by os_inputmain().

```
429                                                                                                 {
430     if(endpoint == 2 || endpoint == -2){
431         if(urbinput[0] != 3)
432             return;
433         urbinput++;
434     }
435     for(int bit = BUTTON_HID_COUNT; bit < N_BUTTONS_HW; bit++){
436         int byte = bit / 8;
437         uchar test = 1 << (bit % 8);
438         if(urbinput[byte] & test)
439             SET_KEYBIT(kbinput, MOUSE_BUTTON_FIRST + bit);
440         else
441             CLEAR_KEYBIT(kbinput, MOUSE_BUTTON_FIRST + bit);
442     }
443 }
```

Here is the caller graph for this function:

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

Definition at line 246 of file keymap.c.

References ckb_warn, CLEAR_KEYBIT, and SET_KEYBIT.

Referenced by os_inputmain().

```
246                                                                                                 {
247     if(length < 1)
248         return;
249     // LUT for HID -> Corsair scancodes (-1 for no scan code, -2 for currently unsupported)
250     // Modified from Linux drivers/hid/usbhid/usbkbd.c, key codes replaced with keymap array indices and
      K95 keys added
251     // Make sure the indices match the keyindex as passed to nprintkey() in notify.c
252     static const short hid_codes[256] = {
253         -1,  -1,  -1,  -1,  37,  54,  52,  39,  27,  40,  41,  42,  32,  43,  44,  45,
254         56,  55,  33,  34,  25,  28,  38,  29,  31,  53,  26,  51,  30,  50,  13,  14,
255         15,  16,  17,  18,  19,  20,  21,  22,  82,   0,  86,  24,  64,  23,  84,  35,
256         79,  80,  81,  46,  47,  12,  57,  58,  59,  36,   1,   2,   3,   4,   5,   6,
257          7,   8,   9,  10,  11,  72,  73,  74,  75,  76,  77,  78,  87,  88,  89,  95,
258         93,  94,  92, 102, 103, 104, 105, 106, 107, 115, 116, 117, 112, 113, 114, 108,
259        109, 110, 118, 119,  49,  69,  -2,  -2,  -2,  -2,  -2,  -2,  -2,  -2,  -2,  -2,
260         -2,  -2,  -2,  -2,  -2,  -2,  -2,  -2,  98,  -2,  -2,  -2,  -2,  -2,  -2,  97,
261        130, 131,  -1,  -1,  -1,  -2,  -1,  83,  66,  85, 145, 144,  -2,  -1,  -1,  -1,
262         -2,  -2,  -2,  -2,  -2,  -1,  -1,  -1,  -1,  -1,  -1,  -1,  -1,  -1,  -1,  -1,
263         -1,  -1,  -1,  -1,  -1,  -1,  -1,  -1,  -1,  -1,  -1,  -1,  -1,  -1,  -1,  -1,
264         -1,  -1,  -1,  -1,  -1,  -1,  -1,  -1,  -1,  -1,  -1,  -1,  -1,  -1,  -1,  -1,
265         -1,  -1,  -1,  -1,  -1,  -1,  -1,  -1,  -1,  -1,  -1,  -1,  -3,  -1,  -1,  -1,  // <- -3 = non-RGB
```

```
    program key
266         120, 121, 122, 123, 124, 125, 126, 127, 128, 129, 136, 137, 138, 139, 140, 141,
267          60,  48,  62,  61,  91,  90,  67,  68, 142, 143,  99, 101,  -2, 130, 131,  97,
268          -2, 133, 134, 135,  -2,  96,  -2, 132,  -2,  -2,  71,  71,  71,  71,  -1,  -1,
269     };
270     switch(endpoint){
271     case 1:
272     case -1:
273         // EP 1: 6KRO input (RGB and non-RGB)
274         // Clear previous input
275         for(int i = 0; i < 256; i++){
276             if(hid_codes[i] >= 0)
277                 CLEAR_KEYBIT(kbinput, hid_codes[i]);
278         }
279         // Set new input
280         for(int i = 0; i < 8; i++){
281             if((urbinput[0] >> i) & 1)
282                 SET_KEYBIT(kbinput, hid_codes[i + 224]);
283         }
284         for(int i = 2; i < length; i++){
285             if(urbinput[i] > 3){
286                 int scan = hid_codes[urbinput[i]];
287                 if(scan >= 0)
288                     SET_KEYBIT(kbinput, scan);
289                 else
290                     ckb_warn("Got unknown key press %d on EP 1\n", urbinput[i]);
291             }
292         }
293         break;
294     case -2:
295         // EP 2 RGB: NKRO input
296         if(urbinput[0] == 1){
297             // Type 1: standard key
298             if(length != 21)
299                 return;
300             for(int bit = 0; bit < 8; bit++){
301                 if((urbinput[1] >> bit) & 1)
302                     SET_KEYBIT(kbinput, hid_codes[bit + 224]);
303                 else
304                     CLEAR_KEYBIT(kbinput, hid_codes[bit + 224]);
305             }
306             for(int byte = 0; byte < 19; byte++){
307                 char input = urbinput[byte + 2];
308                 for(int bit = 0; bit < 8; bit++){
309                     int keybit = byte * 8 + bit;
310                     int scan = hid_codes[keybit];
311                     if((input >> bit) & 1){
312                         if(scan >= 0)
313                             SET_KEYBIT(kbinput, hid_codes[keybit]);
314                         else
315                             ckb_warn("Got unknown key press %d on EP 2\n", keybit);
316                     } else if(scan >= 0)
317                         CLEAR_KEYBIT(kbinput, hid_codes[keybit]);
318                 }
319             }
320             break;
321         } else if (urbinput[0] == 2)
322             ; // Type 2: media key (implicitly falls through)
323         else
324             break;  // No other known types
325         /* FALLTHRU */
326     case 2:
327         // EP 2 Non-RGB: media keys
328         CLEAR_KEYBIT(kbinput, 97);          // mute
329         CLEAR_KEYBIT(kbinput, 98);          // stop
330         CLEAR_KEYBIT(kbinput, 99);          // prev
331         CLEAR_KEYBIT(kbinput, 100);         // play
332         CLEAR_KEYBIT(kbinput, 101);         // next
333         CLEAR_KEYBIT(kbinput, 130);         // volup
334         CLEAR_KEYBIT(kbinput, 131);         // voldn
335         for(int i = 0; i < length; i++){
336             switch(urbinput[i]){
337             case 181:
338                 SET_KEYBIT(kbinput, 101);   // next
339                 break;
340             case 182:
341                 SET_KEYBIT(kbinput, 99);    // prev
342                 break;
343             case 183:
344                 SET_KEYBIT(kbinput, 98);    // stop
345                 break;
346             case 205:
347                 SET_KEYBIT(kbinput, 100);   // play
348                 break;
349             case 226:
350                 SET_KEYBIT(kbinput, 97);    // mute
351                 break;
```

```
352             case 233:
353                 SET_KEYBIT(kbinput, 130);   // volup
354                 break;
355             case 234:
356                 SET_KEYBIT(kbinput, 131);   // voldn
357                 break;
358             }
359         }
360         break;
361     case 3:
362         // EP 3 non-RGB: NKRO input
363         if(length != 15)
364             return;
365         for(int bit = 0; bit < 8; bit++){
366             if((urbinput[0] >> bit) & 1)
367                 SET_KEYBIT(kbinput, hid_codes[bit + 224]);
368             else
369                 CLEAR_KEYBIT(kbinput, hid_codes[bit + 224]);
370         }
371         for(int byte = 0; byte < 14; byte++){
372             char input = urbinput[byte + 1];
373             for(int bit = 0; bit < 8; bit++){
374                 int keybit = byte * 8 + bit;
375                 int scan = hid_codes[keybit];
376                 if((input >> bit) & 1){
377                     if(scan >= 0)
378                         SET_KEYBIT(kbinput, hid_codes[keybit]);
379                     else
380                         ckb_warn("Got unknown key press %d on EP 3\n", keybit);
381                 } else if(scan >= 0)
382                     CLEAR_KEYBIT(kbinput, hid_codes[keybit]);
383             }
384         }
385         break;
386     }
387 }
```

Here is the caller graph for this function:

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

Definition at line 391 of file keymap.c.

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

Referenced by os_inputmain().

```
391                                                                 {
392     //The HID Input Endpoint on FWv3 is 64 bytes, so we can't check for length.
393     if((endpoint != 2 && endpoint != -2) || (fwversion < 0x300 && length < 10))
394         return;
395     // EP 2: mouse input
396     if(urbinput[0] != 1)
397         return;
398     // Byte 1 = mouse buttons (bitfield)
399     for(int bit = 0; bit < BUTTON_HID_COUNT; bit++){
400         if(urbinput[1] & (1 << bit))
401             SET_KEYBIT(kbinput, MOUSE_BUTTON_FIRST + bit);
402         else
403             CLEAR_KEYBIT(kbinput, MOUSE_BUTTON_FIRST + bit);
404     }
405     // Bytes 5 - 8: movement
406     *xaxis += *(short*)(urbinput + 5);
407     *yaxis += *(short*)(urbinput + 7);
408     // Byte 9: wheel
409     char wheel = urbinput[9];
410     if(wheel > 0)
411         SET_KEYBIT(kbinput, MOUSE_EXTRA_FIRST);         // wheelup
```

```
412     else
413         CLEAR_KEYBIT(kbinput, MOUSE_EXTRA_FIRST);
414     if(wheel < 0)
415         SET_KEYBIT(kbinput, MOUSE_EXTRA_FIRST + 1);     // wheeldn
416     else
417         CLEAR_KEYBIT(kbinput, MOUSE_EXTRA_FIRST + 1);
418 }
```

Here is the caller graph for this function:

9.26.4 Variable Documentation

9.26.4.1 const key keymap[(((152+22+12)+25)+12)]

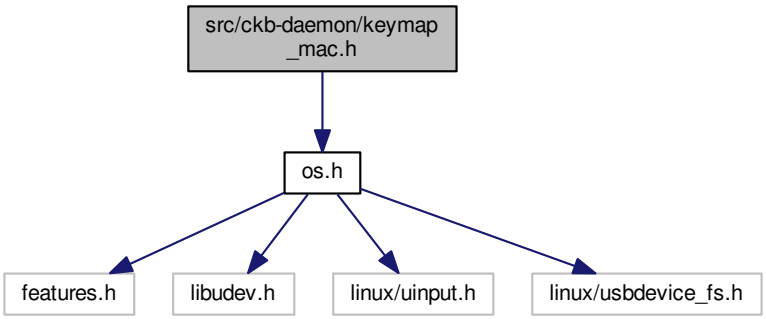
Definition at line 5 of file keymap.c.

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

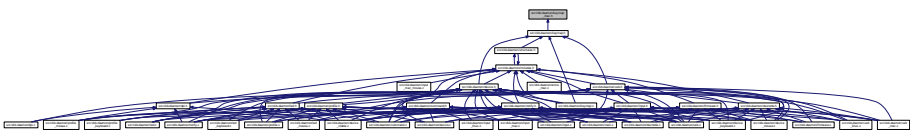
9.27 src/ckb-daemon/keymap_mac.h File Reference

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

Include dependency graph for keymap_mac.h:



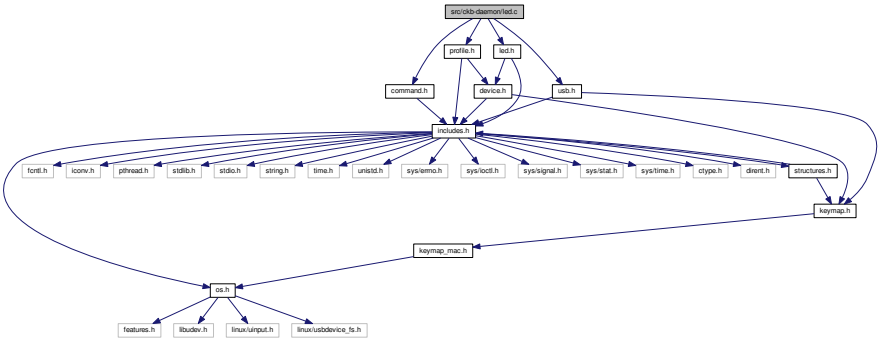
This graph shows which files directly or indirectly include this file:



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

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

Include dependency graph for led.c:



Functions

- void cmd_rgb (usbdevice ∗kb, usbmode ∗mode, int dummy, int keyindex, const char ∗code)
- static uchar iselect (const char ∗led)
- void cmd_ioff (usbdevice ∗kb, usbmode ∗mode, int dummy1, int dummy2, const char ∗led)
- void cmd_ion (usbdevice ∗kb, usbmode ∗mode, int dummy1, int dummy2, const char ∗led)
- void cmd_iauto (usbdevice ∗kb, usbmode ∗mode, int dummy1, int dummy2, const char ∗led)
- void cmd_inotify (usbdevice ∗kb, usbmode ∗mode, int nnumber, int dummy, const char ∗led)
- static int has_key (const char ∗name, const usbdevice ∗kb)
- char ∗ printrgb (const lighting ∗light, const usbdevice ∗kb)

9.28.1 Function Documentation

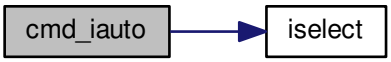
9.28.1.1 void cmd_iauto (usbdevice ∗ *kb,* usbmode ∗ *mode,* int *dummy1,* int *dummy2,* const char ∗ *led*)

Definition at line 63 of file led.c.

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

```
63                                                                                  {
64     (void)dummy1;
65     (void)dummy2;
66
67     uchar bits = iselect(led);
68     // Remove the bits from both ioff and ion
69     mode->ioff &= ~bits;
70     mode->ion &= ~bits;
71     kb->vtable->updateindicators(kb, 0);
72 }
```

Here is the call graph for this function:



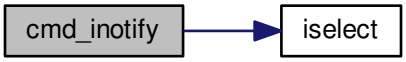
9.28.1.2 void cmd_inotify (usbdevice * *kb*, usbmode * *mode*, int *nnumber*, int *dummy*, const char * *led*)

Definition at line 74 of file led.c.

References usbmode::inotify, and iselect().

```
74                                                                                  {
75     (void)kb;
76     (void)dummy;
77
78     uchar bits = iselect(led);
79     if(strstr(led, ":off"))
80         // Turn notifications for these bits off
81         mode->inotify[nnumber] &= ~bits;
82     else
83         // Turn notifications for these bits on
84         mode->inotify[nnumber] |= bits;
85 }
```

Here is the call graph for this function:



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

Definition at line 41 of file led.c.

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

```
41                                                                                  {
42     (void)dummy1;
43     (void)dummy2;
44
45     uchar bits = iselect(led);
46     // Add the bits to ioff, remove them from ion
47     mode->ioff |= bits;
48     mode->ion &= ~bits;
49     kb->vtable->updateindicators(kb, 0);
50 }
```

Here is the call graph for this function:

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

Definition at line 52 of file led.c.

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

```
52                                                                                  {
53     (void)dummy1;
54     (void)dummy2;
55
56     uchar bits = iselect(led);
57     // Remove the bits from ioff, add them to ion
58     mode->ioff &= ~bits;
59     mode->ion |= bits;
60     kb->vtable->updateindicators(kb, 0);
61 }
```

Here is the call graph for this function:

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

Definition at line 6 of file led.c.

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

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

```
22          mode->light.g[index] = g;
23          mode->light.b[index] = b;
24      }
25  }
```

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

Definition at line 88 of file led.c.

References IS_K63, IS_K65, IS_K95, IS_MOUSE, IS_SABRE, IS_SCIMITAR, usbdevice::product, and usbdevice::vendor.

Referenced by printrgb().

```
88                                                      {
89      if(!name)
90          return 0;
91      if(IS_MOUSE(kb->vendor, kb->product)){
92          // Mice only have the RGB zones
93          if((IS_SABRE(kb) || IS_SCIMITAR(kb)) && !strcmp(name, "wheel"))
94              return 1;
95          if(IS_SCIMITAR(kb) && !strcmp(name, "thumb"))
96              return 1;
97          if(strstr(name, "dpi") == name || !strcmp(name, "front") || !strcmp(name, "back"))
98              return 1;
99          return 0;
100     } else {
101         // But keyboards don't have them at all
102         if(strstr(name, "dpi") == name || !strcmp(name, "front") || !strcmp(name, "back") || !strcmp(name,
    "wheel") || !strcmp(name, "thumb"))
103             return 0;
104         // Only K95 has G keys and M keys (G1 - G18, MR, M1 - M3)
105         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'))))
106             return 0;
107         // K65 and K63 have lights on VolUp/VolDn
108         if((!IS_K65(kb) && !IS_K63(kb)) && (!strcmp(name, "volup") || !strcmp(name, "voldn")))
109             return 0;
110         // K65 lacks numpad and media buttons
111         if(IS_K65(kb) && (strstr(name, "num") == name || !strcmp(name, "stop") || !strcmp(name, "prev
    ") || !strcmp(name, "play") || !strcmp(name, "next")))
112             return 0;
113         // K63 lacks numpad
114         if(IS_K63(kb) && strstr(name, "num") == name)
115             return 0;
116     }
117     return 1;
118 }
```

Here is the caller graph for this function:

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

Definition at line 28 of file led.c.

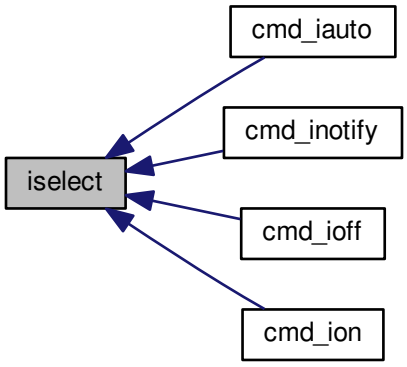
References I_CAPS, I_NUM, and I_SCROLL.

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

```
28                                      {
```

```
29     int result = 0;
30     if(!strncmp(led, "num", 3) || strstr(led, ",num"))
31         result |= I_NUM;
32     if(!strncmp(led, "caps", 4) || strstr(led, ",caps"))
33         result |= I_CAPS;
34     if(!strncmp(led, "scroll", 6) || strstr(led, ",scroll"))
35         result |= I_SCROLL;
36     if(!strncmp(led, "all", 3) || strstr(led, ",all"))
37         result |= I_NUM | I_CAPS | I_SCROLL;
38     return result;
39 }
```

Here is the caller graph for this function:



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

Definition at line 120 of file led.c.

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

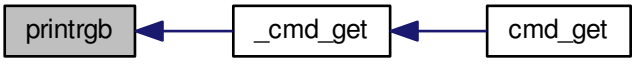
Referenced by _cmd_get().

```
120                                                         {
121     uchar r[N_KEYS_EXTENDED], g[N_KEYS_EXTENDED], b[
    N_KEYS_EXTENDED];
122     const uchar* mr = light->r;
123     const uchar* mg = light->g;
124     const uchar* mb = light->b;
125     for(int i = 0; i < N_KEYS_EXTENDED; i++){
126         // Translate the key index to an RGB index using the key map
127         int k = keymap[i].led;
128         if(k < 0)
129             continue;
130         r[i] = mr[k];
131         g[i] = mg[k];
132         b[i] = mb[k];
133     }
134     // Make a buffer to track key names and to filter out duplicates
135     char names[N_KEYS_EXTENDED][11];
136     for(int i = 0; i < N_KEYS_EXTENDED; i++){
137         const char* name = keymap[i].name;
138         if(keymap[i].led < 0 || !has_key(name, kb))
139             names[i][0] = 0;
140         else
141             strncpy(names[i], name, 11);
142     }
143     // Check to make sure these aren't all the same color
144     int same = 1;
145     for(int i = 1; i < N_KEYS_EXTENDED; i++){
146         if(!names[i][0])
```

```
147             continue;
148         if(r[i] != r[0] || g[i] != g[0] || b[i] != b[0]){
149             same = 0;
150             break;
151         }
152     }
153     // If they are, just output that color
154     if(same){
155         char* buffer = malloc(7);
156         snprintf(buffer, 7, "%02x%02x%02x", r[0], g[0], b[0]);
157         return buffer;
158     }
159     const int BUFFER_LEN = 4096;    // Should be more than enough to fit all keys
160     char* buffer = malloc(BUFFER_LEN);
161     int length = 0;
162     for(int i = 0; i < N_KEYS_EXTENDED; i++){
163         if(!names[i][0])
164             continue;
165         // Print the key name
166         int newlen = 0;
167         snprintf(buffer + length, BUFFER_LEN - length, length == 0 ? "%s%n" : " %s%n", names[i], &newlen);
168         length += newlen;
169         // Look ahead to see if any other keys have this color. If so, print them here as well.
170         uchar kr = r[i], kg = g[i], kb = b[i];
171         for(int j = i + 1; j < N_KEYS_EXTENDED; j++){
172             if(!names[j][0])
173                 continue;
174             if(r[j] != kr || g[j] != kg || b[j] != kb)
175                 continue;
176             snprintf(buffer + length, BUFFER_LEN - length, ",%s%n", names[j], &newlen);
177             length += newlen;
178             // Erase the key's name so it won't get printed later
179             names[j][0] = 0;
180         }
181         // Print the color
182         snprintf(buffer + length, BUFFER_LEN - length, ":%02x%02x%02x%n", kr, kg, kb, &newlen);
183         length += newlen;
184     }
185     return buffer;
186 }
```

Here is the call graph for this function:

Here is the caller graph for this function:

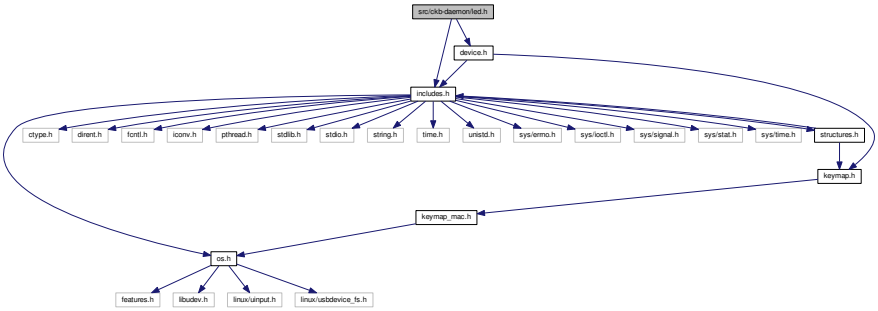


9.29 src/ckb-daemon/led.h File Reference

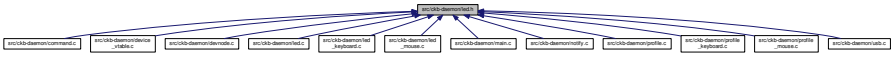
```
#include "includes.h"
```

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

Include dependency graph for led.h:



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



Functions

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

9.29.1 Function Documentation

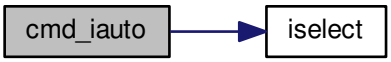
9.29.1.1 void cmd_iauto (usbdevice ∗ *kb,* usbmode ∗ *mode,* int *dummy1,* int *dummy2,* const char ∗ *led*)

Definition at line 63 of file led.c.

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

```
63                                                                                  {
64     (void)dummy1;
65     (void)dummy2;
66
67     uchar bits = iselect(led);
68     // Remove the bits from both ioff and ion
69     mode->ioff &= ~bits;
70     mode->ion &= ~bits;
71     kb->vtable->updateindicators(kb, 0);
72 }
```

Here is the call graph for this function:



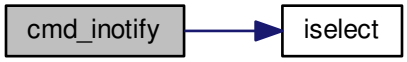
9.29.1.2 void cmd_inotify (usbdevice * *kb*, usbmode * *mode*, int *nnumber*, int *dummy*, const char * *led*)

Definition at line 74 of file led.c.

References usbmode::inotify, and iselect().

```
74                                                                                  {
75     (void)kb;
76     (void)dummy;
77
78     uchar bits = iselect(led);
79     if(strstr(led, ":off"))
80         // Turn notifications for these bits off
81         mode->inotify[nnumber] &= ~bits;
82     else
83         // Turn notifications for these bits on
84         mode->inotify[nnumber] |= bits;
85 }
```

Here is the call graph for this function:



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

Definition at line 41 of file led.c.

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

```
41                                                                                  {
42     (void)dummy1;
43     (void)dummy2;
44
45     uchar bits = iselect(led);
46     // Add the bits to ioff, remove them from ion
47     mode->ioff |= bits;
48     mode->ion &= ~bits;
49     kb->vtable->updateindicators(kb, 0);
50 }
```

Here is the call graph for this function:

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

Definition at line 52 of file led.c.

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

```
52                                                                                  {
53     (void)dummy1;
54     (void)dummy2;
55
56     uchar bits = iselect(led);
57     // Remove the bits from ioff, add them to ion
58     mode->ioff &= ~bits;
59     mode->ion |= bits;
60     kb->vtable->updateindicators(kb, 0);
61 }
```

Here is the call graph for this function:

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

Definition at line 6 of file led.c.

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

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

```
22          mode->light.g[index] = g;
23          mode->light.b[index] = b;
24      }
25  }
```

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

Since Firmware Version 2.05 for K95RGB 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. Up to now we know, that K70RGB Pro and K70 Lux RGB have firmware version 2.04 and having the problem also. So we have to determine in the most inner loop the firmware version and type of KB to select the correct compare-table.

Read colors

```
< That is the old comparison method: you get back what you sent.
```

Normally a firmware version >= 2.05 runs with the new compare array. Up to now there is a 2.04 running in K70 RGB Lux with the same behavior. It seems that K70RGB has the same problem

Definition at line 190 of file led_keyboard.c.

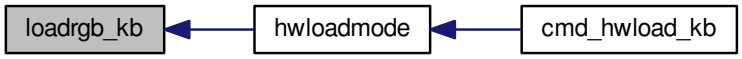
References lighting::b, ckb_err, usbdevice::fwversion, lighting::g, IS_V2_OVERRIDE, MSG_SIZE, N_KEYS_HW, P_K70_LUX, P_K70_LUX_NRGB, usbdevice::product, lighting::r, usbrecv, and usbsend.

Referenced by hwloadmode().

```
190                                                     {
191     if(kb->fwversion >= 0x0120 || IS_V2_OVERRIDE(kb)){
192         uchar data_pkt[12][MSG_SIZE] = {
193             { 0x0e, 0x14, 0x03, 0x01, 0x01, mode + 1, 0x01 },
194             { 0xff, 0x01, 60, 0 },
195             { 0xff, 0x02, 60, 0 },
196             { 0xff, 0x03, 24, 0 },
197             { 0x0e, 0x14, 0x03, 0x01, 0x01, mode + 1, 0x02 },
198             { 0xff, 0x01, 60, 0 },
199             { 0xff, 0x02, 60, 0 },
200             { 0xff, 0x03, 24, 0 },
201             { 0x0e, 0x14, 0x03, 0x01, 0x01, mode + 1, 0x03 },
202             { 0xff, 0x01, 60, 0 },
203             { 0xff, 0x02, 60, 0 },
204             { 0xff, 0x03, 24, 0 },
205         };
206         uchar in_pkt[4][MSG_SIZE] = {
207             { 0x0e, 0x14, 0x03, 0x01 },
208             { 0xff, 0x01, 60, 0 },
209             { 0xff, 0x02, 60, 0 },
210             { 0xff, 0x03, 24, 0 },
211         };
212
218
219         uchar cmp_pkt[4][4] = {
220             { 0x0e, 0x14, 0x03, 0x01 },
221             { 0x0e, 0xff, 0x01, 60 },
222             { 0x0e, 0xff, 0x02, 60 },
223             { 0x0e, 0xff, 0x03, 24 },
224         };
226         uchar* colors[3] = { light->r, light->g, light->b };
227         for(int clr = 0; clr < 3; clr++){
228             for(int i = 0; i < 4; i++){
229                 if(!usbrecv(kb, data_pkt[i + clr * 4], in_pkt[i]))
230                     return -1;
231
232                 uchar* comparePacket = data_pkt[i + clr * 4];
233                 if ((kb->fwversion >= 0x205)
237                         || ((kb->fwversion >= 0x204)
238                             && ((kb->product == P_K70_LUX_NRGB) || (kb->
    product == P_K70_LUX)))) {
239                     comparePacket = cmp_pkt[i];
240                 }
241
242                 if (memcmp(in_pkt[i], comparePacket, 4)) {
243                     ckb_err("Bad input header\n");
244                     ckb_err("color = %d, i = %d, mode = %d\nOutput (Request): %2.2x %2.2x %2.2x
     %2.2x\nInput(Reply): %2.2x %2.2x %2.2x %2.2x %2.2x %2.2x %2.2x %2.2x\n", clr, i, mode,
245                         comparePacket[0], comparePacket[1], comparePacket[2], comparePacket[3],
246                         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]);
247                   in_pkt[2][0] = 0x99;
248                   in_pkt[2][1] = 0x99;
249                   in_pkt[2][2] = 0x99;
250                   in_pkt[2][3] = 0x99;
251                   usbrecv(kb, in_pkt[2], in_pkt[2]); // just to find it in the wireshark log
252                   return -1;
253               }
254           }
255           // Copy colors to lighting. in_pkt[0] is irrelevant.
256           memcpy(colors[clr], in_pkt[1] + 4, 60);
257           memcpy(colors[clr] + 60, in_pkt[2] + 4, 60);
258           memcpy(colors[clr] + 120, in_pkt[3] + 4, 24);
259       }
260    } else {
261       uchar data_pkt[5][MSG_SIZE] = {
262           { 0x0e, 0x14, 0x02, 0x01, 0x01, mode + 1, 0 },
263           { 0xff, 0x01, 60, 0 },
264           { 0xff, 0x02, 60, 0 },
265           { 0xff, 0x03, 60, 0 },
266           { 0xff, 0x04, 36, 0 },
267       };
268       uchar in_pkt[4][MSG_SIZE] = {
269           { 0xff, 0x01, 60, 0 },
270           { 0xff, 0x02, 60, 0 },
271           { 0xff, 0x03, 60, 0 },
272           { 0xff, 0x04, 36, 0 },
273       };
274       // Write initial packet
275       if(!usbsend(kb, data_pkt[0], 1))
276           return -1;
277       // Read colors
278       for(int i = 1; i < 5; i++){
279           if(!usbrecv(kb, data_pkt[i],in_pkt[i - 1]))
280               return -1;
281           if(memcmp(in_pkt[i - 1], data_pkt[i], 4)){
282               ckb_err("Bad input header\n");
283               return -1;
284           }
285       }
286       // Copy the data back to the mode
287       uint8_t mr[N_KEYS_HW / 2], mg[N_KEYS_HW / 2], mb[
     N_KEYS_HW / 2];
288       memcpy(mr,      in_pkt[0] +  4, 60);
289       memcpy(mr + 60, in_pkt[1] +  4, 12);
290       memcpy(mg,      in_pkt[1] + 16, 48);
291       memcpy(mg + 48, in_pkt[2] +  4, 24);
292       memcpy(mb,      in_pkt[2] + 28, 36);
293       memcpy(mb + 36, in_pkt[3] +  4, 36);
294       // Unpack LED data to 8bpc format
295       for(int i = 0; i < N_KEYS_HW; i++){
296           int     i_2 = i / 2;
297           uint8_t r, g, b;
298
299           // 3-bit intensities stored in alternate nybbles.
300           if (i & 1) {
301               r = 7 - (mr[i_2] >> 4);
302               g = 7 - (mg[i_2] >> 4);
303               b = 7 - (mb[i_2] >> 4);
304           } else {
305               r = 7 - (mr[i_2] & 0x0F);
306               g = 7 - (mg[i_2] & 0x0F);
307               b = 7 - (mb[i_2] & 0x0F);
308           }
309           // Scale 3-bit values up to 8 bits.
310           light->r[i] = r << 5 | r << 2 | r >> 1;
311           light->g[i] = g << 5 | g << 2 | g >> 1;
312           light->b[i] = b << 5 | b << 2 | b >> 1;
313       }
314    }
315    return 0;
316 }
```

Here is the caller graph for this function:



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

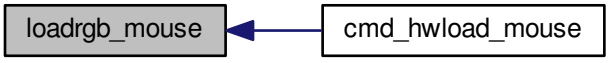
Definition at line 87 of file led_mouse.c.

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().

```
87                                                          {
88     (void)mode;
89
90     uchar data_pkt[MSG_SIZE] = { 0x0e, 0x13, 0x10, 1, 0 };
91     uchar in_pkt[MSG_SIZE] = { 0 };
92     // Load each RGB zone
93     int zonecount = IS_SCIMITAR(kb) ? 4 : IS_SABRE(kb) ? 3 : 2;
94     for(int i = 0; i < zonecount; i++){
95         if(!usbrecv(kb, data_pkt, in_pkt))
96             return -1;
97         if(memcmp(in_pkt, data_pkt, 4)){
98             ckb_err("Bad input header\n");
99             return -2;
100         }
101         // Copy data
102         int led = LED_MOUSE + i;
103         if(led >= LED_DPI)
104             led++;          // Skip DPI light
105         light->r[led] = in_pkt[4];
106         light->g[led] = in_pkt[5];
107         light->b[led] = in_pkt[6];
108         // Set packet for next zone
109         data_pkt[2]++;
110     }
111     return 0;
112 }
```

Here is the caller graph for this function:



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

Definition at line 120 of file led.c.

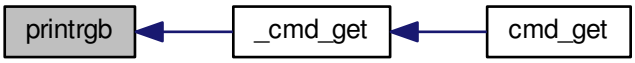
References lighting::b, lighting::g, has_key(), keymap, key::led, N_KEYS_EXTENDED, key::name, and lighting::r.

Referenced by _cmd_get().

```
120                                                     {
121     uchar r[N_KEYS_EXTENDED], g[N_KEYS_EXTENDED], b[
    N_KEYS_EXTENDED];
122     const uchar* mr = light->r;
123     const uchar* mg = light->g;
124     const uchar* mb = light->b;
125     for(int i = 0; i < N_KEYS_EXTENDED; i++){
126         // Translate the key index to an RGB index using the key map
127         int k = keymap[i].led;
128         if(k < 0)
129             continue;
130         r[i] = mr[k];
131         g[i] = mg[k];
132         b[i] = mb[k];
133     }
134     // Make a buffer to track key names and to filter out duplicates
135     char names[N_KEYS_EXTENDED][11];
136     for(int i = 0; i < N_KEYS_EXTENDED; i++){
137         const char* name = keymap[i].name;
138         if(keymap[i].led < 0 || !has_key(name, kb))
139             names[i][0] = 0;
140         else
141             strncpy(names[i], name, 11);
142     }
143     // Check to make sure these aren't all the same color
144     int same = 1;
145     for(int i = 1; i < N_KEYS_EXTENDED; i++){
146         if(!names[i][0])
147             continue;
148         if(r[i] != r[0] || g[i] != g[0] || b[i] != b[0]){
149             same = 0;
150             break;
151         }
152     }
153     // If they are, just output that color
154     if(same){
155         char* buffer = malloc(7);
156         snprintf(buffer, 7, "%02x%02x%02x", r[0], g[0], b[0]);
157         return buffer;
158     }
159     const int BUFFER_LEN = 4096;    // Should be more than enough to fit all keys
160     char* buffer = malloc(BUFFER_LEN);
161     int length = 0;
162     for(int i = 0; i < N_KEYS_EXTENDED; i++){
163         if(!names[i][0])
164             continue;
165         // Print the key name
166         int newlen = 0;
167         snprintf(buffer + length, BUFFER_LEN - length, length == 0 ? "%s%n" : " %s%n", names[i], &newlen);
168         length += newlen;
169         // Look ahead to see if any other keys have this color. If so, print them here as well.
170         uchar kr = r[i], kg = g[i], kb = b[i];
171         for(int j = i + 1; j < N_KEYS_EXTENDED; j++){
172             if(!names[j][0])
173                 continue;
174             if(r[j] != kr || g[j] != kg || b[j] != kb)
175                 continue;
176             snprintf(buffer + length, BUFFER_LEN - length, ",%s%n", names[j], &newlen);
177             length += newlen;
178             // Erase the key's name so it won't get printed later
179             names[j][0] = 0;
180         }
181         // Print the color
182         snprintf(buffer + length, BUFFER_LEN - length, ":%02x%02x%02x%n", kr, kg, kb, &newlen);
183         length += newlen;
184     }
185     return buffer;
186 }
```

Here is the call graph for this function:

Here is the caller graph for this function:



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

Definition at line 148 of file led_keyboard.c.

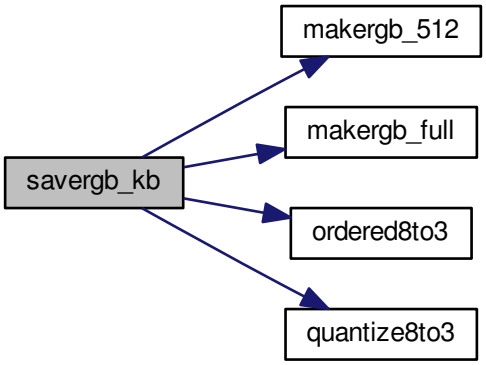
References usbdevice::dither, usbdevice::fwversion, IS_STRAFE, IS_V2_OVERRIDE, makergb_512(), makergb_-full(), MSG_SIZE, ordered8to3(), quantize8to3(), and usbsend.

Referenced by cmd_hwsave_kb().

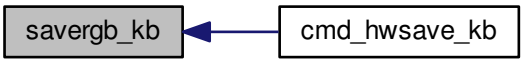
```
148                                                     {
149     if(kb->fwversion >= 0x0120 || IS_V2_OVERRIDE(kb)){
150         uchar data_pkt[12][MSG_SIZE] = {
151             // Red
152             { 0x7f, 0x01, 60, 0 },
153             { 0x7f, 0x02, 60, 0 },
154             { 0x7f, 0x03, 24, 0 },
155             { 0x07, 0x14, 0x03, 0x01, 0x01, mode + 1, 0x01 },
156             // Green
157             { 0x7f, 0x01, 60, 0 },
158             { 0x7f, 0x02, 60, 0 },
159             { 0x7f, 0x03, 24, 0 },
160             { 0x07, 0x14, 0x03, 0x01, 0x01, mode + 1, 0x02 },
161             // Blue
162             { 0x7f, 0x01, 60, 0 },
163             { 0x7f, 0x02, 60, 0 },
164             { 0x7f, 0x03, 24, 0 },
165             { 0x07, 0x14, 0x03, 0x01, 0x01, mode + 1, 0x03 }
166         };
167         makergb_full(light, data_pkt);
168         if(!usbsend(kb, data_pkt[0], 12))
169             return -1;
170         if (IS_STRAFE(kb)){ // end save
171             uchar save_end_pkt[MSG_SIZE] = { 0x07, 0x14, 0x04, 0x01, 0x01 };
172             if(!usbsend(kb, save_end_pkt, 1))
173                 return -1;
174         }
175     } else {
176         uchar data_pkt[5][MSG_SIZE] = {
177             { 0x7f, 0x01, 60, 0 },
178             { 0x7f, 0x02, 60, 0 },
179             { 0x7f, 0x03, 60, 0 },
180             { 0x7f, 0x04, 36, 0 },
181             { 0x07, 0x14, 0x02, 0x00, 0x01, mode + 1 }
182         };
```

```
183         makergb_512(light, data_pkt, kb->dither ? ordered8to3 :
    quantize8to3);
184         if(!usbsend(kb, data_pkt[0], 5))
185             return -1;
186     }
187     return 0;
188 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



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

Definition at line 66 of file led_mouse.c.

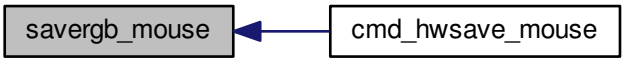
References lighting::b, lighting::g, IS_SABRE, IS_SCIMITAR, LED_DPI, LED_MOUSE, MSG_SIZE, lighting::r, and usbsend.

Referenced by cmd_hwsave_mouse().

```
66                                                      {
67     (void)mode;
68
69     uchar data_pkt[MSG_SIZE] = { 0x07, 0x13, 0x10, 1, 0 };
70     // Save each RGB zone, minus the DPI light which is sent in the DPI packets
71     int zonecount = IS_SCIMITAR(kb) ? 4 : IS_SABRE(kb) ? 3 : 2;
72     for(int i = 0; i < zonecount; i++){
73         int led = LED_MOUSE + i;
74         if(led >= LED_DPI)
75             led++;          // Skip DPI light
76         data_pkt[4] = light->r[led];
```

```
77          data_pkt[5] = light->g[led];
78          data_pkt[6] = light->b[led];
79          if(!usbsend(kb, data_pkt, 1))
80              return -1;
81          // Set packet for next zone
82          data_pkt[2]++;
83      }
84      return 0;
85  }
```

Here is the caller graph for this function:



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

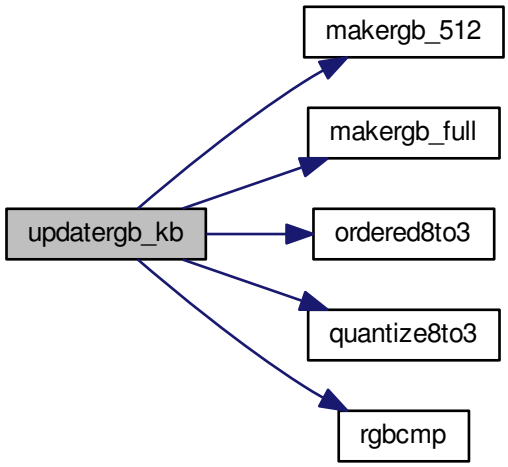
Definition at line 79 of file led_keyboard.c.

References usbdevice::active, usbprofile::currentmode, usbdevice::dither, lighting::forceupdate, IS_FULLRANGE, usbprofile::lastlight, usbmode::light, makergb_512(), makergb_full(), MSG_SIZE, ordered8to3(), P_K95_PLATINUM, usbdevice::product, usbdevice::profile, quantize8to3(), rgbcmp(), lighting::sidelight, and usbsend.

```
79                                          {
80      if(!kb->active)
81          return 0;
82      lighting* lastlight = &kb->profile->lastlight;
83      lighting* newlight = &kb->profile->currentmode->
        light;
84      // Don't do anything if the lighting hasn't changed
85      if(!force && !lastlight->forceupdate && !newlight->forceupdate
86              && !rgbcmp(lastlight, newlight) && lastlight->sidelight == newlight->
        sidelight)   // strafe sidelights
87          return 0;
88      lastlight->forceupdate = newlight->forceupdate = 0;
89
90      if(IS_FULLRANGE(kb)){
91          // Update strafe sidelights if necessary
92          if(lastlight->sidelight != newlight->sidelight) {
93              uchar data_pkt[2][MSG_SIZE] = {
94                   { 0x07, 0x05, 0x08, 0x00, 0x00 },
95                   { 0x07, 0x05, 0x02, 0, 0x03 }
96               };
97               if (newlight->sidelight)
98                   data_pkt[0][4]=1;    // turn on
99               if(!usbsend(kb, data_pkt[0], 2))
100                   return -1;
101          }
102          // 16.8M color lighting works fine on strafe and is the only way it actually works
103          uchar data_pkt[12][MSG_SIZE] = {
104              // Red
105              { 0x7f, 0x01, 0x3c, 0 },
106              { 0x7f, 0x02, 0x3c, 0 },
107              { 0x7f, 0x03, 0x18, 0 },
108              { 0x07, 0x28, 0x01, 0x03, 0x01, 0},
109              // Green
110              { 0x7f, 0x01, 0x3c, 0 },
111              { 0x7f, 0x02, 0x3c, 0 },
112              { 0x7f, 0x03, 0x18, 0 },
113              { 0x07, 0x28, 0x02, 0x03, 0x01, 0},
114              // Blue
115              { 0x7f, 0x01, 0x3c, 0 },
116              { 0x7f, 0x02, 0x3c, 0 },
117              { 0x7f, 0x03, 0x18, 0 },
118              { 0x07, 0x28, 0x03, 0x03, 0x02, 0}
119          };
120          // The K95 Platinum needs 0x30 for the lightbar to work, due to the length of the packet.
```

```
121         // A way to dynamically calculate the length would be preferred, based on the device.
122         if(kb->product == P_K95_PLATINUM){
123             data_pkt[2][2] = 0x30;
124             data_pkt[6][2] = 0x30;
125             data_pkt[10][2] = 0x30;
126         }
127         makergb_full(newlight, data_pkt);
128         if(!usbsend(kb, data_pkt[0], 12))
129             return -1;
130     } else {
131         // On older keyboards it looks flickery and causes lighting glitches, so we don't use it.
132         uchar data_pkt[5][MSG_SIZE] = {
133             { 0x7f, 0x01, 60, 0 },
134             { 0x7f, 0x02, 60, 0 },
135             { 0x7f, 0x03, 60, 0 },
136             { 0x7f, 0x04, 36, 0 },
137             { 0x07, 0x27, 0x00, 0x00, 0xD8 }
138         };
139         makergb_512(newlight, data_pkt, kb->dither ?
      ordered8to3 : quantize8to3);
140         if(!usbsend(kb, data_pkt[0], 5))
141             return -1;
142     }
143
144     memcpy(lastlight, newlight, sizeof(lighting));
145     return 0;
146 }
```

Here is the call graph for this function:



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

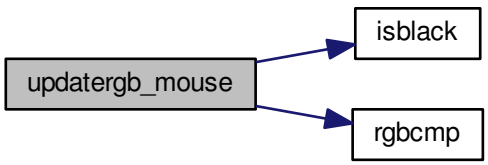
Definition at line 20 of file led_mouse.c.

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

```
20                                          {
21     if(!kb->active)
22         return 0;
23     lighting* lastlight = &kb->profile->lastlight;
24     lighting* newlight = &kb->profile->currentmode->
      light;
25     // Don't do anything if the lighting hasn't changed
```

```
26      if(!force && !lastlight->forceupdate && !newlight->forceupdate
27              && !rgbcmp(lastlight, newlight))
28          return 0;
29      lastlight->forceupdate = newlight->forceupdate = 0;
30
31      // Prevent writing to DPI LEDs or non-existent LED zones for the Glaive.
32      int num_zones = IS_GLAIVE(kb) ? 3 : N_MOUSE_ZONES;
33      // Send the RGB values for each zone to the mouse
34      uchar data_pkt[2][MSG_SIZE] = {
35          { 0x07, 0x22, num_zones, 0x01, 0 }, // RGB colors
36          { 0x07, 0x05, 0x02, 0 }                 // Lighting on/off
37      };
38      uchar* rgb_data = &data_pkt[0][4];
39      for(int i = 0; i < N_MOUSE_ZONES; i++){
40          if (IS_GLAIVE(kb) && i != 0 && i != 1 && i != 5)
41          continue;
42          *rgb_data++ = i + 1;
43          *rgb_data++ = newlight->r[LED_MOUSE + i];
44          *rgb_data++ = newlight->g[LED_MOUSE + i];
45          *rgb_data++ = newlight->b[LED_MOUSE + i];
46      }
47      // Send RGB data
48      if(!usbsend(kb, data_pkt[0], 1))
49          return -1;
50      int was_black = isblack(kb, lastlight), is_black = isblack(kb, newlight);
51      if(is_black){
52          // If the lighting is black, send the deactivation packet (M65 only)
53          if(!usbsend(kb, data_pkt[1], 1))
54              return -1;
55      } else if(was_black || force){
56          // If the lighting WAS black, or if we're on forced update, send the activation packet
57          data_pkt[1][4] = 1;
58          if(!usbsend(kb, data_pkt[1], 1))
59              return -1;
60      }
61
62      memcpy(lastlight, newlight, sizeof(lighting));
63      return 0;
64  }
```

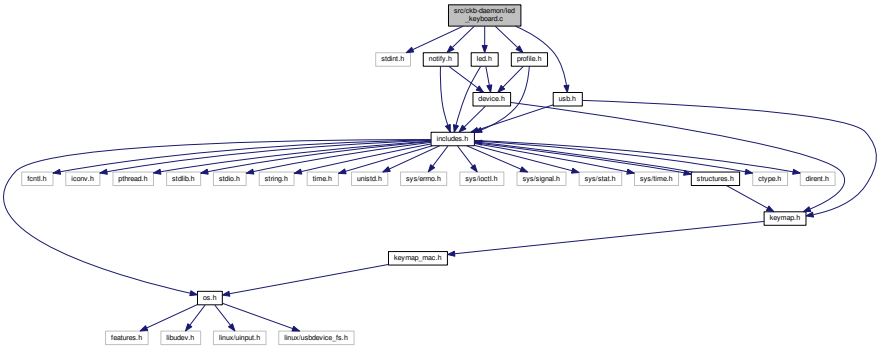
Here is the call graph for this function:



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

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

Include dependency graph for led_keyboard.c:



Macros

- #define BR1(x) ((((x) & 0xaa) >> 1) | (((x) & 0x55) << 1))
- #define BR2(x) (((BR1(x) & 0xcc) >> 2) | ((BR1(x) & 0x33) << 2))
- #define BR4(x) (((BR2(x) & 0xf0) >> 4) | ((BR2(x) & 0x0f) << 4))
- #define O0(i) BR4(i),
- #define O1(i) O0(i) O0((i) + 1)
- #define O2(i) O1(i) O1((i) + 2)
- #define O3(i) O2(i) O2((i) + 4)
- #define O4(i) O3(i) O3((i) + 8)
- #define O5(i) O4(i) O4((i) + 16)
- #define O6(i) O5(i) O5((i) + 32)
- #define O7(i) O6(i) O6((i) + 64)
- #define O8(i) O7(i) O7((i) + 127)

Functions

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

Variables

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

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

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

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

9.30.1 Macro Definition Documentation

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

Definition at line 9 of file led_keyboard.c.

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

Definition at line 10 of file led_keyboard.c.

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

Definition at line 11 of file led_keyboard.c.

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

Definition at line 12 of file led_keyboard.c.

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

Definition at line 13 of file led_keyboard.c.

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

Definition at line 14 of file led_keyboard.c.

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

Definition at line 15 of file led_keyboard.c.

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

Definition at line 16 of file led_keyboard.c.

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

Definition at line 17 of file led_keyboard.c.

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

Definition at line 18 of file led_keyboard.c.

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

Definition at line 19 of file led_keyboard.c.

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

Definition at line 20 of file led_keyboard.c.

9.30.2 Function Documentation

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

Since Firmware Version 2.05 for K95RGB 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. Up to now we know, that K70RGB Pro and K70 Lux RGB have firmware version 2.04 and having the problem also. So we have to determine in the most inner loop the firmware version and type of KB to select the correct compare-table.

Read colors

```
                                                    < That is the old comparison method: you get back what you sent.
```

Normally a firmware version >= 2.05 runs with the new compare array. Up to now there is a 2.04 running in K70 RGB Lux with the same behavior. It seems that K70RGB has the same problem

Definition at line 190 of file led_keyboard.c.

References lighting::b, ckb_err, usbdevice::fwversion, lighting::g, IS_V2_OVERRIDE, MSG_SIZE, N_KEYS_HW, P_K70_LUX, P_K70_LUX_NRGB, usbdevice::product, lighting::r, usbrecv, and usbsend.

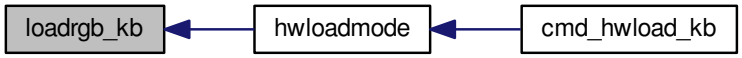
Referenced by hwloadmode().

```
190                                                     {
191     if(kb->fwversion >= 0x0120 || IS_V2_OVERRIDE(kb)){
192         uchar data_pkt[12][MSG_SIZE] = {
193             { 0x0e, 0x14, 0x03, 0x01, 0x01, mode + 1, 0x01 },
```

```
194             { 0xff, 0x01, 60, 0 },
195             { 0xff, 0x02, 60, 0 },
196             { 0xff, 0x03, 24, 0 },
197             { 0x0e, 0x14, 0x03, 0x01, 0x01, mode + 1, 0x02 },
198             { 0xff, 0x01, 60, 0 },
199             { 0xff, 0x02, 60, 0 },
200             { 0xff, 0x03, 24, 0 },
201             { 0x0e, 0x14, 0x03, 0x01, 0x01, mode + 1, 0x03 },
202             { 0xff, 0x01, 60, 0 },
203             { 0xff, 0x02, 60, 0 },
204             { 0xff, 0x03, 24, 0 },
205         };
206         uchar in_pkt[4][MSG_SIZE] = {
207             { 0x0e, 0x14, 0x03, 0x01 },
208             { 0xff, 0x01, 60, 0 },
209             { 0xff, 0x02, 60, 0 },
210             { 0xff, 0x03, 24, 0 },
211         };
212
213
214         uchar cmp_pkt[4][4] = {
215             { 0x0e, 0x14, 0x03, 0x01 },
216             { 0x0e, 0xff, 0x01, 60 },
217             { 0x0e, 0xff, 0x02, 60 },
218             { 0x0e, 0xff, 0x03, 24 },
219         };
220         uchar* colors[3] = { light->r, light->g, light->b };
221         for(int clr = 0; clr < 3; clr++){
222             for(int i = 0; i < 4; i++){
223                 if(!usbrecv(kb, data_pkt[i + clr * 4], in_pkt[i]))
224                     return -1;
225
226                 uchar* comparePacket = data_pkt[i + clr * 4];
227                 if ((kb->fwversion >= 0x205)
228                         || ((kb->fwversion >= 0x204)
229                             && ((kb->product == P_K70_LUX_NRGB) || (kb->
    product == P_K70_LUX)))) {
230                     comparePacket = cmp_pkt[i];
231                 }
232
233                 if (memcmp(in_pkt[i], comparePacket, 4)) {
234                     ckb_err("Bad input header\n");
235                     ckb_err("color = %d, i = %d, mode = %d\nOutput (Request): %2.2x %2.2x %2.2x
     %2.2x\nInput(Reply): %2.2x %2.2x %2.2x %2.2x %2.2x %2.2x %2.2x %2.2x %2.2x\n", clr, i, mode,
236                         comparePacket[0], comparePacket[1], comparePacket[2], comparePacket[3],
237                         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]);
238                     in_pkt[2][0] = 0x99;
239                     in_pkt[2][1] = 0x99;
240                     in_pkt[2][2] = 0x99;
241                     in_pkt[2][3] = 0x99;
242                     usbrecv(kb, in_pkt[2], in_pkt[2]); // just to find it in the wireshark log
243                     return -1;
244                 }
245             }
246             // Copy colors to lighting. in_pkt[0] is irrelevant.
247             memcpy(colors[clr], in_pkt[1] + 4, 60);
248             memcpy(colors[clr] + 60, in_pkt[2] + 4, 60);
249             memcpy(colors[clr] + 120, in_pkt[3] + 4, 24);
250         }
251      } else {
252         uchar data_pkt[5][MSG_SIZE] = {
253             { 0x0e, 0x14, 0x02, 0x01, 0x01, mode + 1, 0 },
254             { 0xff, 0x01, 60, 0 },
255             { 0xff, 0x02, 60, 0 },
256             { 0xff, 0x03, 60, 0 },
257             { 0xff, 0x04, 36, 0 },
258         };
259         uchar in_pkt[4][MSG_SIZE] = {
260             { 0xff, 0x01, 60, 0 },
261             { 0xff, 0x02, 60, 0 },
262             { 0xff, 0x03, 60, 0 },
263             { 0xff, 0x04, 36, 0 },
264         };
265         // Write initial packet
266         if(!usbsend(kb, data_pkt[0], 1))
267             return -1;
268         // Read colors
269         for(int i = 1; i < 5; i++){
270             if(!usbrecv(kb, data_pkt[i],in_pkt[i - 1]))
271                 return -1;
272             if(memcmp(in_pkt[i - 1], data_pkt[i], 4)){
273                 ckb_err("Bad input header\n");
274                 return -1;
275             }
276         }
277         // Copy the data back to the mode
```

```
287         uint8_t mr[N_KEYS_HW / 2], mg[N_KEYS_HW / 2], mb[
    N_KEYS_HW / 2];
288         memcpy(mr,      in_pkt[0] +  4, 60);
289         memcpy(mr + 60, in_pkt[1] +  4, 12);
290         memcpy(mg,      in_pkt[1] + 16, 48);
291         memcpy(mg + 48, in_pkt[2] +  4, 24);
292         memcpy(mb,      in_pkt[2] + 28, 36);
293         memcpy(mb + 36, in_pkt[3] +  4, 36);
294         // Unpack LED data to 8bpc format
295         for(int i = 0; i < N_KEYS_HW; i++){
296             int     i_2 = i / 2;
297             uint8_t r, g, b;
298 
299             // 3-bit intensities stored in alternate nybbles.
300             if (i & 1) {
301                 r = 7 - (mr[i_2] >> 4);
302                 g = 7 - (mg[i_2] >> 4);
303                 b = 7 - (mb[i_2] >> 4);
304             } else {
305                 r = 7 - (mr[i_2] & 0x0F);
306                 g = 7 - (mg[i_2] & 0x0F);
307                 b = 7 - (mb[i_2] & 0x0F);
308             }
309             // Scale 3-bit values up to 8 bits.
310             light->r[i] = r << 5 | r << 2 | r >> 1;
311             light->g[i] = g << 5 | g << 2 | g >> 1;
312             light->b[i] = b << 5 | b << 2 | b >> 1;
313         }
314     }
315     return 0;
316 }
```

Here is the caller graph for this function:



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

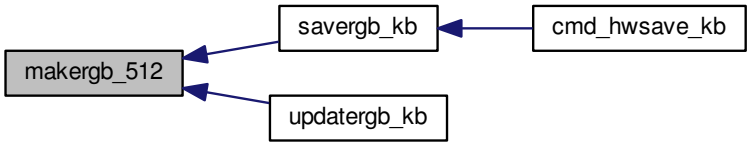
Definition at line 38 of file led_keyboard.c.

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

Referenced by savergb_kb(), and updatergb_kb().

```
39                                                          {
40     uchar r[N_KEYS_HW / 2], g[N_KEYS_HW / 2], b[N_KEYS_HW / 2];
41     // Compress RGB values to a 512-color palette
42     for(int i = 0; i < N_KEYS_HW; i += 2){
43         char r1 = ditherfn(i, light->r[i]), r2 = ditherfn(i + 1, light->r[i + 1]);
44         char g1 = ditherfn(i, light->g[i]), g2 = ditherfn(i + 1, light->g[i + 1]);
45         char b1 = ditherfn(i, light->b[i]), b2 = ditherfn(i + 1, light->b[i + 1]);
46         r[i / 2] = (7 - r2) << 4 | (7 - r1);
47         g[i / 2] = (7 - g2) << 4 | (7 - g1);
48         b[i / 2] = (7 - b2) << 4 | (7 - b1);
49     }
50     memcpy(data_pkt[0] + 4, r, 60);
51     memcpy(data_pkt[1] + 4, r + 60, 12);
52     memcpy(data_pkt[1] + 16, g, 48);
53     memcpy(data_pkt[2] + 4, g + 48, 24);
54     memcpy(data_pkt[2] + 28, b, 36);
55     memcpy(data_pkt[3] + 4, b + 36, 36);
56 }
```

Here is the caller graph for this function:



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

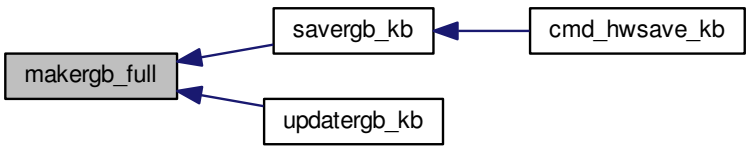
Definition at line 58 of file led_keyboard.c.

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

Referenced by savergb_kb(), and updatergb_kb().

```
58                                                                                  {
59     const uchar* r = light->r, *g = light->g, *b = light->b;
60     // Red
61     memcpy(data_pkt[0] + 4, r, 60);
62     memcpy(data_pkt[1] + 4, r + 60, 60);
63     memcpy(data_pkt[2] + 4, r + 120, 60);
64     // Green (final R packet is blank)
65     memcpy(data_pkt[4] + 4, g, 60);
66     memcpy(data_pkt[5] + 4, g + 60, 60);
67     memcpy(data_pkt[6] + 4, g + 120, 60);
68     // Blue (final G packet is blank)
69     memcpy(data_pkt[8] + 4, b, 60);
70     memcpy(data_pkt[9] + 4, b + 60, 60);
71     memcpy(data_pkt[10] + 4, b + 120, 60);
72 }
```

Here is the caller graph for this function:



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

Definition at line 24 of file led_keyboard.c.

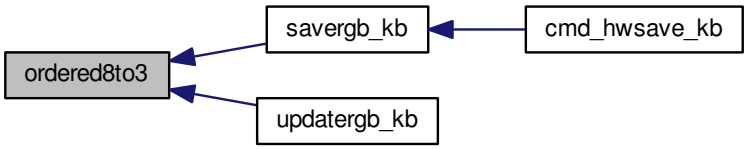
References bit_reverse_table.

Referenced by savergb_kb(), and updatergb_kb().

```
24                                                  {
```

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

Here is the caller graph for this function:



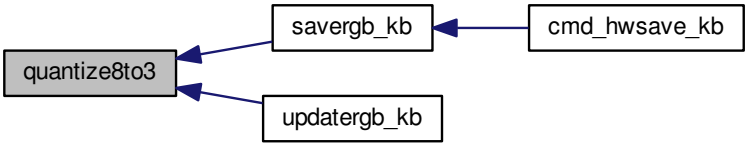
9.30.2.5 static uchar quantize8to3 (int *index*, uchar *value*) [static]

Definition at line 32 of file led_keyboard.c.

Referenced by savergb_kb(), and updatergb_kb().

```
32                                                  {
33     (void)index;
34
35     return value >> 5;
36 }
```

Here is the caller graph for this function:



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

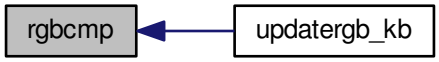
Definition at line 74 of file led_keyboard.c.

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

Referenced by updatergb_kb().

```
74                                                          {
75     // Compare two light structures, ignore mouse zones
76     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);
77 }
```

Here is the caller graph for this function:



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

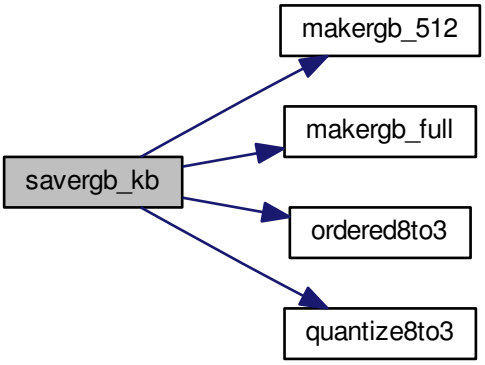
Definition at line 148 of file led_keyboard.c.

References usbdevice::dither, usbdevice::fwversion, IS_STRAFE, IS_V2_OVERRIDE, makergb_512(), makergb_full(), MSG_SIZE, ordered8to3(), quantize8to3(), and usbsend.

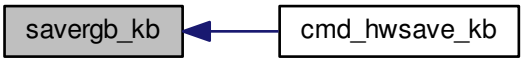
Referenced by cmd_hwsave_kb().

```
148                                                 {
149     if(kb->fwversion >= 0x0120 || IS_V2_OVERRIDE(kb)){
150         uchar data_pkt[12][MSG_SIZE] = {
151             // Red
152             { 0x7f, 0x01, 60, 0 },
153             { 0x7f, 0x02, 60, 0 },
154             { 0x7f, 0x03, 24, 0 },
155             { 0x07, 0x14, 0x03, 0x01, 0x01, mode + 1, 0x01 },
156             // Green
157             { 0x7f, 0x01, 60, 0 },
158             { 0x7f, 0x02, 60, 0 },
159             { 0x7f, 0x03, 24, 0 },
160             { 0x07, 0x14, 0x03, 0x01, 0x01, mode + 1, 0x02 },
161             // Blue
162             { 0x7f, 0x01, 60, 0 },
163             { 0x7f, 0x02, 60, 0 },
164             { 0x7f, 0x03, 24, 0 },
165             { 0x07, 0x14, 0x03, 0x01, 0x01, mode + 1, 0x03 }
166         };
167         makergb_full(light, data_pkt);
168         if(!usbsend(kb, data_pkt[0], 12))
169             return -1;
170         if (IS_STRAFE(kb)){ // end save
171             uchar save_end_pkt[MSG_SIZE] = { 0x07, 0x14, 0x04, 0x01, 0x01 };
172             if(!usbsend(kb, save_end_pkt, 1))
173                 return -1;
174         }
175     } else {
176         uchar data_pkt[5][MSG_SIZE] = {
177             { 0x7f, 0x01, 60, 0 },
178             { 0x7f, 0x02, 60, 0 },
179             { 0x7f, 0x03, 60, 0 },
180             { 0x7f, 0x04, 36, 0 },
181             { 0x07, 0x14, 0x02, 0x00, 0x01, mode + 1 }
182         };
183         makergb_512(light, data_pkt, kb->dither ? ordered8to3 :
    quantize8to3);
184         if(!usbsend(kb, data_pkt[0], 5))
185             return -1;
186     }
187     return 0;
188 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



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

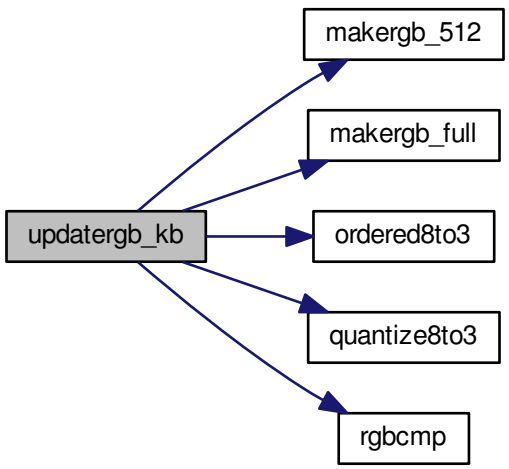
Definition at line 79 of file led_keyboard.c.

References usbdevice::active, usbprofile::currentmode, usbdevice::dither, lighting::forceupdate, IS_FULLRANGE, usbprofile::lastlight, usbmode::light, makergb_512(), makergb_full(), MSG_SIZE, ordered8to3(), P_K95_PLATINUM, usbdevice::product, usbdevice::profile, quantize8to3(), rgbcmp(), lighting::sidelight, and usbsend.

```
79                                          {
80     if(!kb->active)
81         return 0;
82     lighting* lastlight = &kb->profile->lastlight;
83     lighting* newlight = &kb->profile->currentmode->
       light;
84     // Don't do anything if the lighting hasn't changed
85     if(!force && !lastlight->forceupdate && !newlight->forceupdate
86             && !rgbcmp(lastlight, newlight) && lastlight->sidelight == newlight->
       sidelight)   // strafe sidelights
87         return 0;
88     lastlight->forceupdate = newlight->forceupdate = 0;
89
90     if(IS_FULLRANGE(kb)){
91         // Update strafe sidelights if necessary
92         if(lastlight->sidelight != newlight->sidelight) {
93             uchar data_pkt[2][MSG_SIZE] = {
94                 { 0x07, 0x05, 0x08, 0x00, 0x00 },
95                 { 0x07, 0x05, 0x02, 0, 0x03 }
96              };
97              if (newlight->sidelight)
```

```
98                  data_pkt[0][4]=1;    // turn on
99              if(!usbsend(kb, data_pkt[0], 2))
100                 return -1;
101         }
102         // 16.8M color lighting works fine on strafe and is the only way it actually works
103         uchar data_pkt[12][MSG_SIZE] = {
104             // Red
105             { 0x7f, 0x01, 0x3c, 0 },
106             { 0x7f, 0x02, 0x3c, 0 },
107             { 0x7f, 0x03, 0x18, 0 },
108             { 0x07, 0x28, 0x01, 0x03, 0x01, 0},
109             // Green
110             { 0x7f, 0x01, 0x3c, 0 },
111             { 0x7f, 0x02, 0x3c, 0 },
112             { 0x7f, 0x03, 0x18, 0 },
113             { 0x07, 0x28, 0x02, 0x03, 0x01, 0},
114             // Blue
115             { 0x7f, 0x01, 0x3c, 0 },
116             { 0x7f, 0x02, 0x3c, 0 },
117             { 0x7f, 0x03, 0x18, 0 },
118             { 0x07, 0x28, 0x03, 0x03, 0x02, 0}
119         };
120         // The K95 Platinum needs 0x30 for the lightbar to work, due to the length of the packet.
121         // A way to dynamically calculate the length would be preferred, based on the device.
122         if(kb->product == P_K95_PLATINUM){
123             data_pkt[2][2] = 0x30;
124             data_pkt[6][2] = 0x30;
125             data_pkt[10][2] = 0x30;
126         }
127         makergb_full(newlight, data_pkt);
128         if(!usbsend(kb, data_pkt[0], 12))
129             return -1;
130     } else {
131         // On older keyboards it looks flickery and causes lighting glitches, so we don't use it.
132         uchar data_pkt[5][MSG_SIZE] = {
133             { 0x7f, 0x01, 60, 0 },
134             { 0x7f, 0x02, 60, 0 },
135             { 0x7f, 0x03, 60, 0 },
136             { 0x7f, 0x04, 36, 0 },
137             { 0x07, 0x27, 0x00, 0x00, 0xD8 }
138         };
139         makergb_512(newlight, data_pkt, kb->dither ?
    ordered8to3 : quantize8to3);
140         if(!usbsend(kb, data_pkt[0], 5))
141             return -1;
142     }
143
144     memcpy(lastlight, newlight, sizeof(lighting));
145     return 0;
146 }
```

Here is the call graph for this function:



9.30.3 Variable Documentation

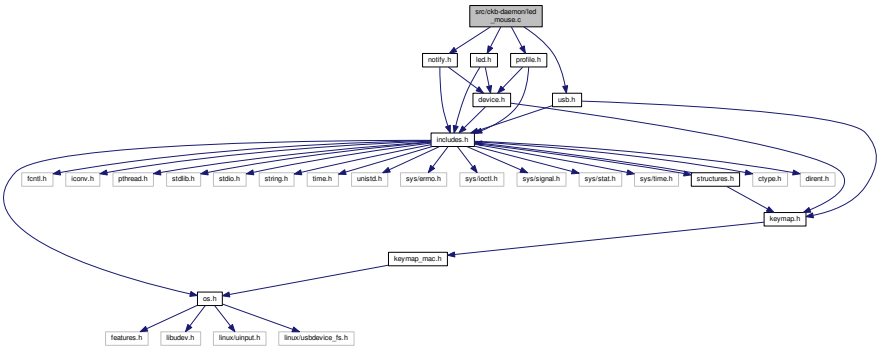
9.30.3.1 uchar bit_reverse_table[256] = { (((((((((0) & 0xaa) >> 1) | ((((0) & 0x55) << 1)) & 0xcc) >> 2) | (((((0) & 0xaa) >> 1) | ((((0) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((0) & 0xaa) >> 1) | ((((0) & 0x55) << 1)) & 0xcc) >> 2) | (((((0) & 0xaa) >> 1) | ((((0) & 0x55) << 1)) & 0x33)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

Referenced by ordered8to3().

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

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

Include dependency graph for led_mouse.c:



Functions

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

9.31.1 Function Documentation

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

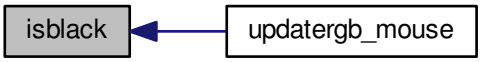
Definition at line 13 of file led_mouse.c.

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

Referenced by updatergb_mouse().

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

Here is the caller graph for this function:



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

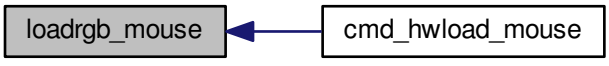
Definition at line 87 of file led_mouse.c.

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().

```
87                                                                  {
88      (void)mode;
89
90      uchar data_pkt[MSG_SIZE] = { 0x0e, 0x13, 0x10, 1, 0 };
91      uchar in_pkt[MSG_SIZE] = { 0 };
92      // Load each RGB zone
93      int zonecount = IS_SCIMITAR(kb) ? 4 : IS_SABRE(kb) ? 3 : 2;
94      for(int i = 0; i < zonecount; i++){
95          if(!usbrecv(kb, data_pkt, in_pkt))
96              return -1;
97          if(memcmp(in_pkt, data_pkt, 4)){
98              ckb_err("Bad input header\n");
99              return -2;
100         }
101         // Copy data
102         int led = LED_MOUSE + i;
103         if(led >= LED_DPI)
104             led++;          // Skip DPI light
105         light->r[led] = in_pkt[4];
106         light->g[led] = in_pkt[5];
107         light->b[led] = in_pkt[6];
108         // Set packet for next zone
109         data_pkt[2]++;
110     }
111     return 0;
112 }
```

Here is the caller graph for this function:



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

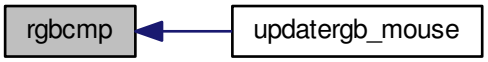
Definition at line 7 of file led_mouse.c.

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

Referenced by updatergb_mouse().

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

Here is the caller graph for this function:



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

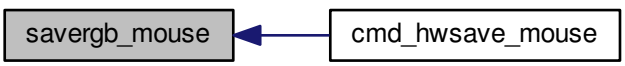
Definition at line 66 of file led_mouse.c.

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

Referenced by cmd_hwsave_mouse().

```
66                                                                  {
67      (void)mode;
68
69      uchar data_pkt[MSG_SIZE] = { 0x07, 0x13, 0x10, 1, 0 };
70      // Save each RGB zone, minus the DPI light which is sent in the DPI packets
71      int zonecount = IS_SCIMITAR(kb) ? 4 : IS_SABRE(kb) ? 3 : 2;
72      for(int i = 0; i < zonecount; i++){
73          int led = LED_MOUSE + i;
74          if(led >= LED_DPI)
75              led++;          // Skip DPI light
76          data_pkt[4] = light->r[led];
77          data_pkt[5] = light->g[led];
78          data_pkt[6] = light->b[led];
79          if(!usbsend(kb, data_pkt, 1))
80              return -1;
81          // Set packet for next zone
82          data_pkt[2]++;
83      }
84      return 0;
85 }
```

Here is the caller graph for this function:



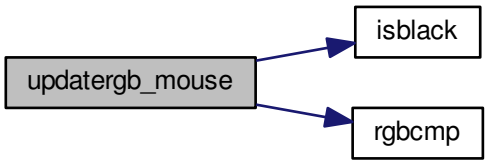
9.31.1.5 int updatergb_mouse (usbdevice * *kb*, int *force*)

Definition at line 20 of file led_mouse.c.

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

```
20                                              {
21     if(!kb->active)
22         return 0;
23     lighting* lastlight = &kb->profile->lastlight;
24     lighting* newlight = &kb->profile->currentmode->
       light;
25     // Don't do anything if the lighting hasn't changed
26     if(!force && !lastlight->forceupdate && !newlight->forceupdate
27             && !rgbcmp(lastlight, newlight))
28         return 0;
29     lastlight->forceupdate = newlight->forceupdate = 0;
30
31     // Prevent writing to DPI LEDs or non-existent LED zones for the Glaive.
32     int num_zones = IS_GLAIVE(kb) ? 3 : N_MOUSE_ZONES;
33     // Send the RGB values for each zone to the mouse
34     uchar data_pkt[2][MSG_SIZE] = {
35         { 0x07, 0x22, num_zones, 0x01, 0 }, // RGB colors
36         { 0x07, 0x05, 0x02, 0 }                 // Lighting on/off
37     };
38     uchar* rgb_data = &data_pkt[0][4];
39     for(int i = 0; i < N_MOUSE_ZONES; i++){
40         if (IS_GLAIVE(kb) && i != 0 && i != 1 && i != 5)
41         continue;
42         *rgb_data++ = i + 1;
43         *rgb_data++ = newlight->r[LED_MOUSE + i];
44         *rgb_data++ = newlight->g[LED_MOUSE + i];
45         *rgb_data++ = newlight->b[LED_MOUSE + i];
46     }
47     // Send RGB data
48     if(!usbsend(kb, data_pkt[0], 1))
49         return -1;
50     int was_black = isblack(kb, lastlight), is_black = isblack(kb, newlight);
51     if(is_black){
52         // If the lighting is black, send the deactivation packet (M65 only)
53         if(!usbsend(kb, data_pkt[1], 1))
54             return -1;
55     } else if(was_black || force){
56         // If the lighting WAS black, or if we're on forced update, send the activation packet
57         data_pkt[1][4] = 1;
58         if(!usbsend(kb, data_pkt[1], 1))
59             return -1;
60     }
61
62     memcpy(lastlight, newlight, sizeof(lighting));
63     return 0;
64 }
```

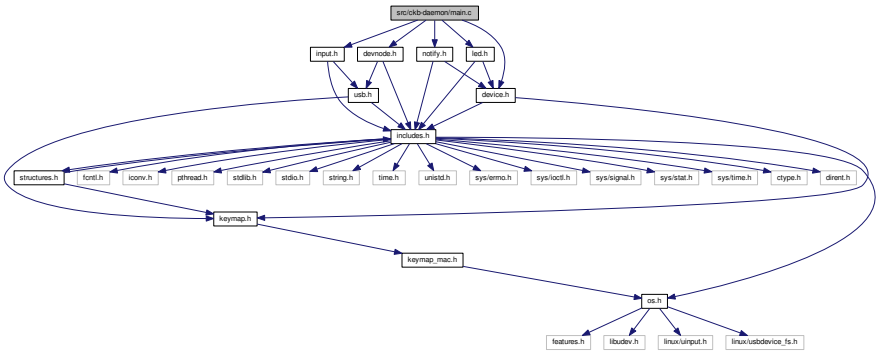
Here is the call graph for this function:



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

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

Include dependency graph for main.c:



Functions

- static void quitWithLock (char mut)

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

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

Variables

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

  brief .
- int features_mask

  brief .
- int hwload_mode

  hwload_mode = 1 means read hardware once. should be enough

9.32.1 Function Documentation

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

Definition at line 71 of file main.c.

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

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

Definition at line 88 of file main.c.

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

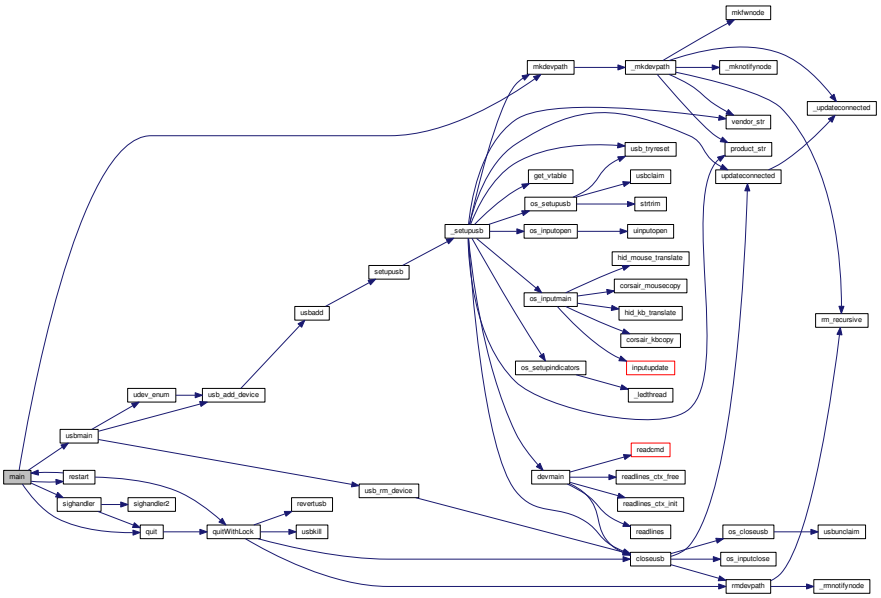
Referenced by restart().

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

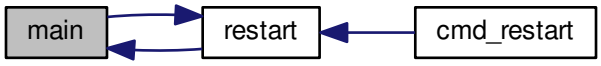
```
136     FILE* pidfile = fopen(pidpath, "r");
137     if(pidfile){
138         pid_t pid;
139         fscanf(pidfile, "%d", &pid);
140         fclose(pidfile);
141         if(pid > 0){
142             // kill -s 0 checks if the PID is active but doesn't send a signal
143             if(!kill(pid, 0)){
144                 ckb_fatal_nofile("ckb-daemon is already running (PID %d). Try `killall
    ckb-daemon`.\n", pid);
145                 ckb_fatal_nofile("(If you're certain the process is dead, delete %s and try
    again)\n", pidpath);
146                 return 0;
147             }
148         }
149     }
150 
151     // Read parameters
152     int forceroot = 1;
153     for(int i = 1; i < argc; i++){
154         char* argument = argv[i];
155         unsigned newgid;
156         char hwload[7];
157         if(sscanf(argument, "--gid=%u", &newgid) == 1){
158             // Set dev node GID
159             gid = newgid;
160             ckb_info_nofile("Setting /dev node gid: %u\n", newgid);
161         } else if(!strcmp(argument, "--nobind")){
162             // Disable key notifications and rebinding
163             features_mask &= ~FEAT_BIND & ~FEAT_NOTIFY;
164             ckb_info_nofile("Key binding and key notifications are disabled\n");
165         } else if(!strcmp(argument, "--nonotify")){
166             // Disable key notifications
167             features_mask &= ~FEAT_NOTIFY;
168             ckb_info_nofile("Key notifications are disabled\n");
169         } else if(sscanf(argument, "--hwload=%6s", hwload) == 1){
170             if(!strcmp(hwload, "always") || !strcmp(hwload, "yes") || !strcmp(hwload, "y") || !strcmp(
    hwload, "a")){
171                 hwload_mode = 2;
172                 ckb_info_nofile("Setting hardware load: always\n");
173             } else if(!strcmp(hwload, "tryonce") || !strcmp(hwload, "try") || !strcmp(hwload, "once") || !
    strcmp(hwload, "t") || !strcmp(hwload, "o")){
174                 hwload_mode = 1;
175                 ckb_info_nofile("Setting hardware load: tryonce\n");
176             } else if(!strcmp(hwload, "never") || !strcmp(hwload, "none") || !strcmp(hwload, "no") || !
    strcmp(hwload, "n")){
177                 hwload_mode = 0;
178                 ckb_info_nofile("Setting hardware load: never\n");
179             }
180         } else if(!strcmp(argument, "--nonroot")){
181             // Allow running as a non-root user
182             forceroot = 0;
183         }
184 #ifdef OS_MAC
185         else if(!strcmp(argument, "--nomouseaccel")){
186             // On OSX, provide an option to disable mouse acceleration
187             features_mask &= ~FEAT_MOUSEACCEL;
188             ckb_info_nofile("Mouse acceleration disabled\n");
189         }
190 #endif
191     }
192 
193     // Check UID
194     if(getuid() != 0){
195         if(forceroot){
196             ckb_fatal_nofile("ckb-daemon must be run as root. Try `sudo %s`\n", argv[0]);
197             exit(0);
198         } else
199             ckb_warn_nofile("Warning: not running as root, allowing anyway per command-line
    parameter...\n");
200     }
201 
202     // Make root keyboard
203     umask(0);
204     memset(keyboard, 0, sizeof(keyboard));
205     if(!mkdevpath(keyboard))
206         ckb_info("Root controller ready at %s0\n", devpath);
207 
208     // Set signals
209     sigset_t signals;
210     sigfillset(&signals);
211     sigdelset(&signals, SIGTERM);
212     sigdelset(&signals, SIGINT);
213     sigdelset(&signals, SIGQUIT);
214     sigdelset(&signals, SIGUSR1);
215     // Set up signal handlers for quitting the service.
216     sigprocmask(SIG_SETMASK, &signals, 0);
```

```
217     signal(SIGTERM, sighandler);
218     signal(SIGINT, sighandler);
219     signal(SIGQUIT, sighandler);
220     signal(SIGUSR1, (void (*)())restart);
221
222     // Start the USB system
223     int result = usbmain();
224     quit();
225     return result;
226 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



9.32.1.3 static void quit ( ) [static]

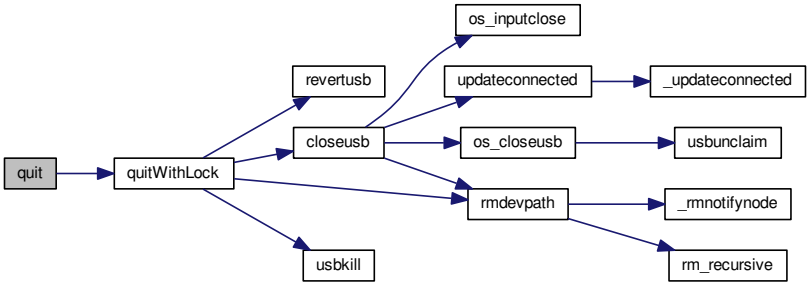
Definition at line 30 of file main.c.

References quitWithLock().

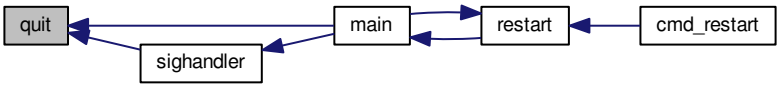
Referenced by main(), and sighandler().

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

Here is the call graph for this function:



Here is the caller graph for this function:



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

Parameters

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

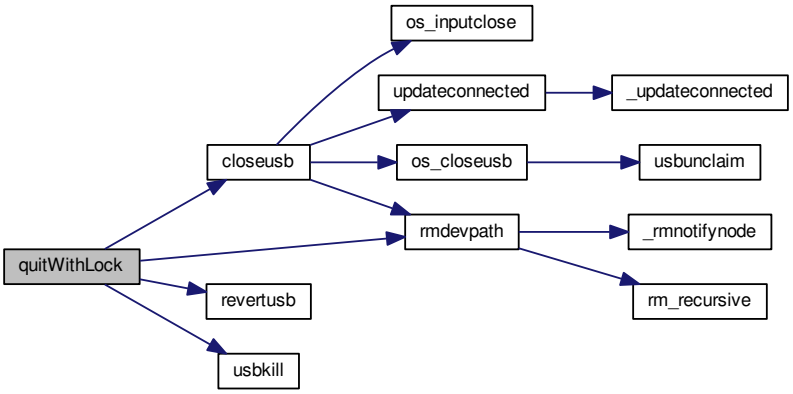
Definition at line 40 of file main.c.

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

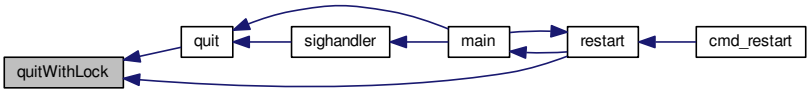
Referenced by quit(), and restart().

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

Here is the call graph for this function:



Here is the caller graph for this function:



9.32.1.5 int restart ( )

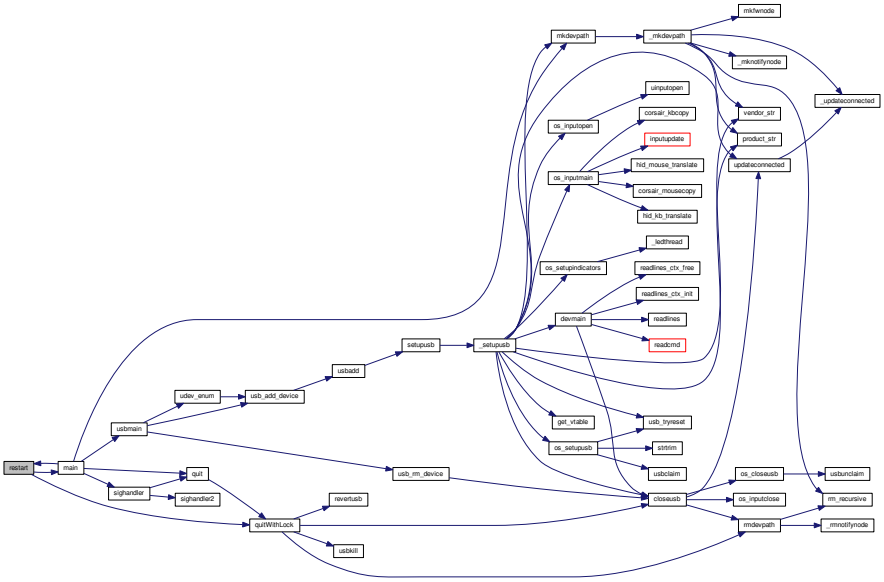
Definition at line 228 of file main.c.

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

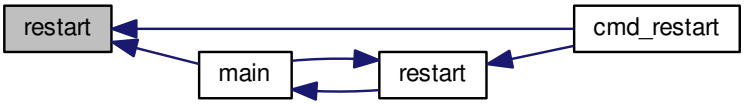
Referenced by cmd_restart(), and main().

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

Here is the call graph for this function:



Here is the caller graph for this function:



9.32.1.6 void sighandler (int *type*)

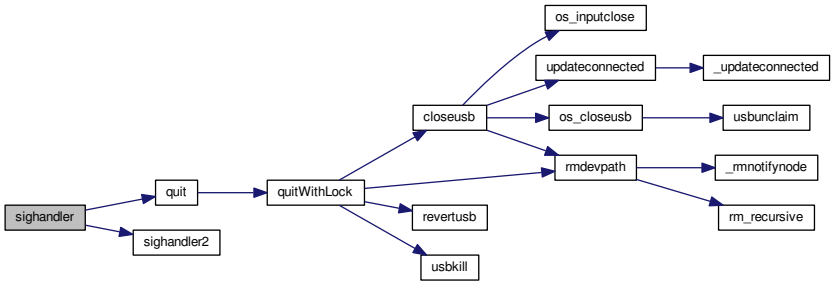
Definition at line 62 of file main.c.

References quit(), and sighandler2().

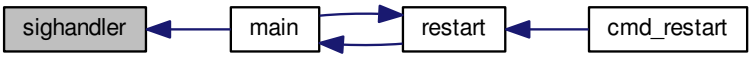
Referenced by main().

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

Here is the call graph for this function:



Here is the caller graph for this function:



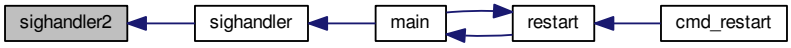
9.32.1.7 void sighandler2 (int *type*)

Definition at line 57 of file main.c.

Referenced by sighandler().

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

Here is the caller graph for this function:



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

Definition at line 19 of file main.c.

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

9.32.2 Variable Documentation

9.32.2.1 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().

9.32.2.2 int hwload_mode

Definition at line 7 of file device.c.

Referenced by main().

9.32.2.3 int main_ac [static]

Definition at line 7 of file main.c.

Referenced by main(), and restart().

9.32.2.4 char** main_av [static]

Definition at line 8 of file main.c.

Referenced by main(), and restart().

9.32.2.5 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.

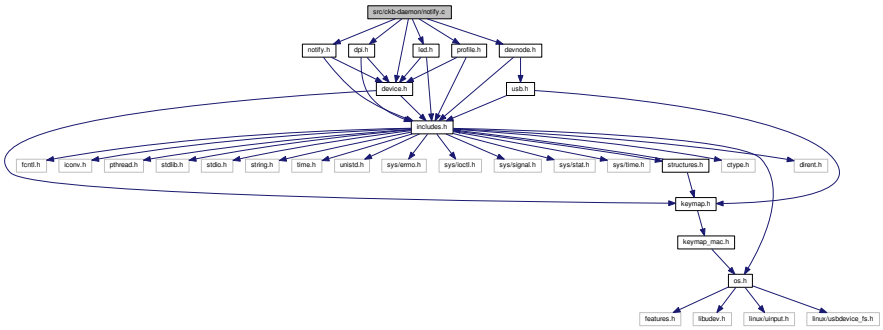
Definition at line 25 of file usb.c.

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

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

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

Include dependency graph for notify.c:



Macros

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

Functions

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

9.33.1 Macro Definition Documentation

9.33.1.1 #define HW_STANDARD

Value:

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

Definition at line 83 of file notify.c.

Referenced by _cmd_get().

9.33.1.2 #define HWMODE_OR_RETURN(*kb, index*)

Value:

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

Definition at line 73 of file notify.c.

9.33.2 Function Documentation

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

Definition at line 90 of file notify.c.

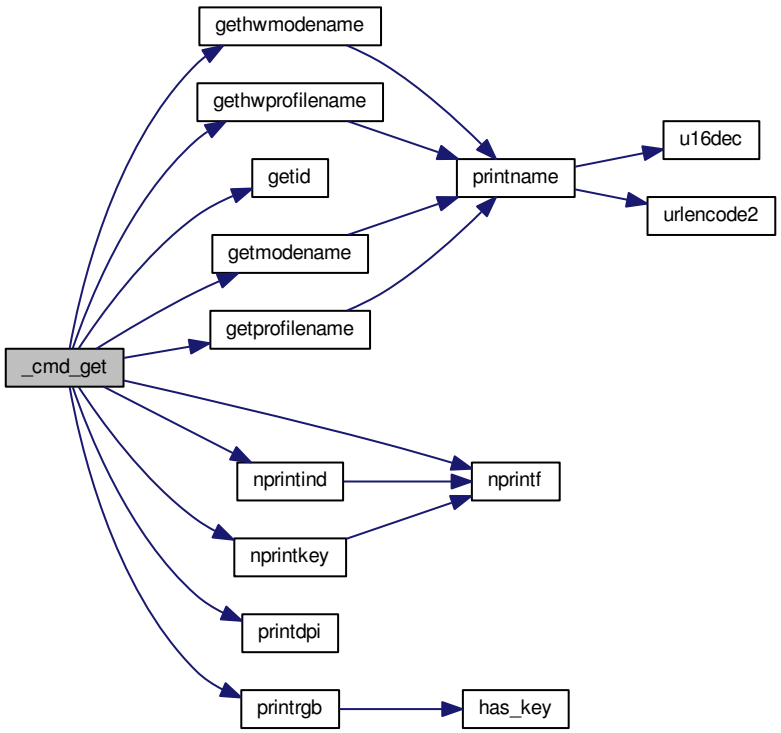
References dpiset::current, usbmode::dpi, hwprofile::dpi, gethwmodename(), gethwprofilename(), getid(), getmodename(), getprofilename(), usbdevice::hw, usbdevice::hw_ileds, HW_STANDARD, I_CAPS, I_NUM, I_SCROLL, usbmode::id, usbprofile::id, hwprofile::id, usbdevice::input, keymap, usbinput::keys, dpiset::lift,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

Referenced by cmd_get().

```
90                                                                                                  {
91      usbprofile* profile = kb->profile;
92      if(!strcmp(setting, ":mode")){
93          // Get the current mode number
94          nprintf(kb, nnumber, mode, "switch\n");
95          return;
96      } else if(!strcmp(setting, ":rgb")){
97          // Get the current RGB settings
98          char* rgb = printrgb(&mode->light, kb);
99          nprintf(kb, nnumber, mode, "rgb %s\n", rgb);
100          free(rgb);
101          return;
102      } else if(!strcmp(setting, ":hwrgb")){
103          // Get the current hardware RGB settings
104          HW_STANDARD;
105          char* rgb = printrgb(kb->hw->light + index, kb);
106          nprintf(kb, nnumber, mode, "hwrgb %s\n", rgb);
107          free(rgb);
108          return;
109      } else if(!strcmp(setting, ":profilename")){
110          // Get the current profile name
111          char* name = getprofilename(profile);
112          nprintf(kb, nnumber, 0, "profilename %s\n", name[0] ? name : "Unnamed");
113          free(name);
114      } else if(!strcmp(setting, ":name")){
115          // Get the current mode name
116          char* name = getmodename(mode);
117          nprintf(kb, nnumber, mode, "name %s\n", name[0] ? name : "Unnamed");
118          free(name);
119      } else if(!strcmp(setting, ":hwprofilename")){
120          // Get the current hardware profile name
121          if(!kb->hw)
122              return;
123          char* name = gethwprofilename(kb->hw);
124          nprintf(kb, nnumber, 0, "hwprofilename %s\n", name[0] ? name : "Unnamed");
125          free(name);
126      } else if(!strcmp(setting, ":hwname")){
127          // Get the current hardware mode name
128          HW_STANDARD;
129          char* name = gethwmodename(kb->hw, index);
130          nprintf(kb, nnumber, mode, "hwname %s\n", name[0] ? name : "Unnamed");
131          free(name);
132      } else if(!strcmp(setting, ":profileid")){
133          // Get the current profile ID
134          char* guid = getid(&profile->id);
135          int modified;
136          memcpy(&modified, &profile->id.modified, sizeof(modified));
137          nprintf(kb, nnumber, 0, "profileid %s %x\n", guid, modified);
138          free(guid);
139      } else if(!strcmp(setting, ":id")){
140          // Get the current mode ID
141          char* guid = getid(&mode->id);
142          int modified;
143          memcpy(&modified, &mode->id.modified, sizeof(modified));
144          nprintf(kb, nnumber, mode, "id %s %x\n", guid, modified);
145          free(guid);
146      } else if(!strcmp(setting, ":hwprofileid")){
147          // Get the current hardware profile ID
148          if(!kb->hw)
149              return;
150          char* guid = getid(&kb->hw->id[0]);
151          int modified;
152          memcpy(&modified, &kb->hw->id[0].modified, sizeof(modified));
153          nprintf(kb, nnumber, 0, "hwprofileid %s %x\n", guid, modified);
```

```
154         free(guid);
155     } else if(!strcmp(setting, ":hwid")){
156         // Get the current hardware mode ID
157         HW_STANDARD;
158         char* guid = getid(&kb->hw->id[index + 1]);
159         int modified;
160         memcpy(&modified, &kb->hw->id[index + 1].modified, sizeof(modified));
161         nprintf(kb, nnumber, mode, "hwid %s %x\n", guid, modified);
162         free(guid);
163     } else if(!strcmp(setting, ":keys")){
164         // Get the current state of all keys
165         for(int i = 0; i < N_KEYS_INPUT; i++){
166             if(!keymap[i].name)
167                 continue;
168             int byte = i / 8, bit = 1 << (i & 7);
169             uchar state = kb->input.keys[byte] & bit;
170             if(state)
171                 nprintkey(kb, nnumber, i, 1);
172         }
173     } else if(!strcmp(setting, ":i")){
174         // Get the current state of all indicator LEDs
175         if(kb->hw_ileds & I_NUM) nprintind(kb, nnumber,
    I_NUM, 1);
176         if(kb->hw_ileds & I_CAPS) nprintind(kb, nnumber,
    I_CAPS, 1);
177         if(kb->hw_ileds & I_SCROLL) nprintind(kb, nnumber,
    I_SCROLL, 1);
178     } else if(!strcmp(setting, ":dpi")){
179         // Get the current DPI levels
180         char* dpi = printdpi(&mode->dpi, kb);
181         nprintf(kb, nnumber, mode, "dpi %s\n", dpi);
182         free(dpi);
183         return;
184     } else if(!strcmp(setting, ":hwdpi")){
185         // Get the current hardware DPI levels
186         HW_STANDARD;
187         char* dpi = printdpi(kb->hw->dpi + index, kb);
188         nprintf(kb, nnumber, mode, "hwdpi %s\n", dpi);
189         free(dpi);
190         return;
191     } else if(!strcmp(setting, ":dpisel")){
192         // Get the currently-selected DPI
193         nprintf(kb, nnumber, mode, "dpisel %d\n", mode->dpi.current);
194     } else if(!strcmp(setting, ":hwdpisel")){
195         // Get the currently-selected hardware DPI
196         HW_STANDARD;
197         nprintf(kb, nnumber, mode, "hwdpisel %d\n", kb->hw->dpi[index].
    current);
198     } else if(!strcmp(setting, ":lift")){
199         // Get the mouse lift height
200         nprintf(kb, nnumber, mode, "lift %d\n", mode->dpi.lift);
201     } else if(!strcmp(setting, ":hwlift")){
202         // Get the hardware lift height
203         HW_STANDARD;
204         nprintf(kb, nnumber, mode, "hwlift %d\n", kb->hw->dpi[index].
    lift);
205     } else if(!strcmp(setting, ":snap")){
206         // Get the angle snap status
207         nprintf(kb, nnumber, mode, "snap %s\n", mode->dpi.snap ? "on" : "off");
208     } else if(!strcmp(setting, ":hwsnap")){
209         // Get the hardware angle snap status
210         HW_STANDARD;
211         nprintf(kb, nnumber, mode, "hwsnap %s\n", kb->hw->dpi[index].
    snap ? "on" : "off");
212     }
213 }
```

Here is the call graph for this function:



Here is the caller graph for this function:

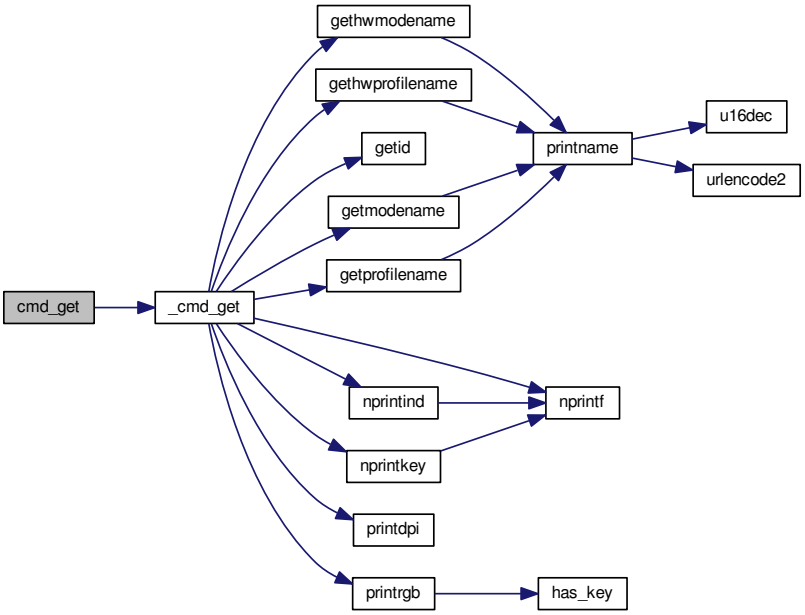
9.33.2.2 void cmd_get (usbdevice * *kb*, usbmode * *mode*, int *nnumber*, int *dummy*, const char * *setting*)

Definition at line 215 of file notify.c.

References _cmd_get(), and imutex.

```
215                                                                                 {
216     (void)dummy;
217
218     pthread_mutex_lock(imutex(kb));
219     _cmd_get(kb, mode, nnumber, setting);
220     pthread_mutex_unlock(imutex(kb));
221 }
```

Here is the call graph for this function:



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

Definition at line 61 of file notify.c.

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

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

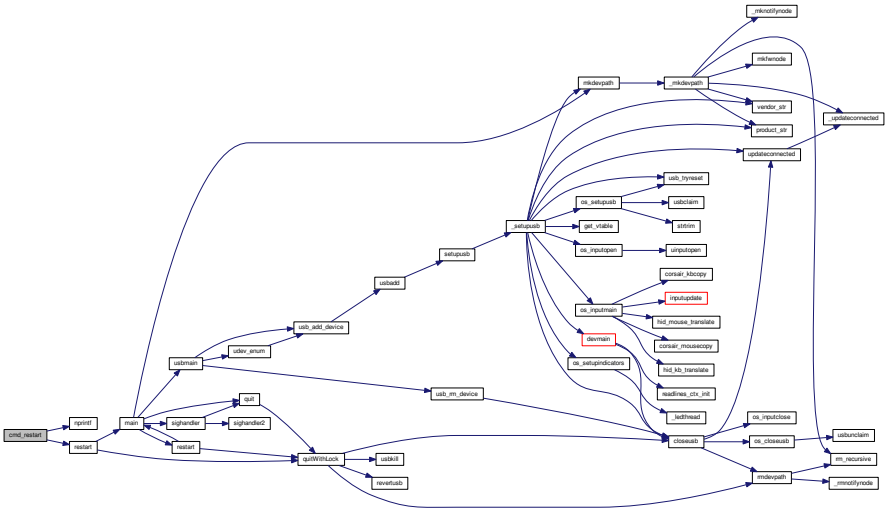
9.33.2.4 void cmd_restart (usbdevice * *kb*, usbmode * *mode*, int *nnumber*, int *dummy*, const char * *content*)

Definition at line 225 of file notify.c.

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

```
225                                                                                                 {
226     (void)mode;
227     (void)nnumber;
228     (void)dummy;
229
230     ckb_info("RESTART called with %s\n", content);
231     nprintf(kb, -1, 0, "RESTART called with %s\n", content);
232     restart();
233 }
```

Here is the call graph for this function:



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

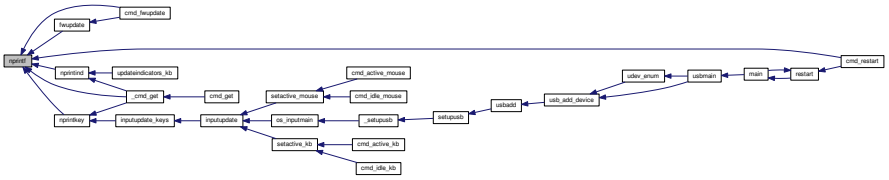
Definition at line 8 of file notify.c.

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

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

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

Here is the caller graph for this function:



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

Definition at line 43 of file notify.c.

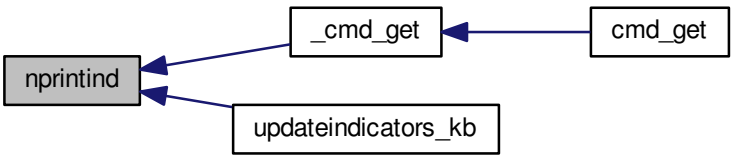
References I_CAPS, I_NUM, I_SCROLL, and nprintf().

Referenced by _cmd_get(), and updateindicators_kb().

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

Here is the call graph for this function:

Here is the caller graph for this function:



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

Definition at line 35 of file notify.c.

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

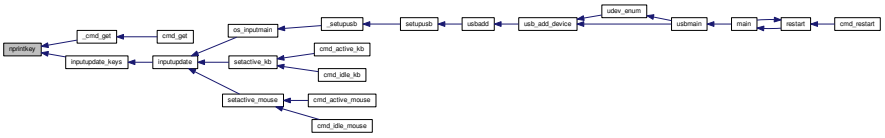
Referenced by _cmd_get(), and inputupdate_keys().

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

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

Here is the call graph for this function:

Here is the caller graph for this function:



9.33.2.8 int restart ( )

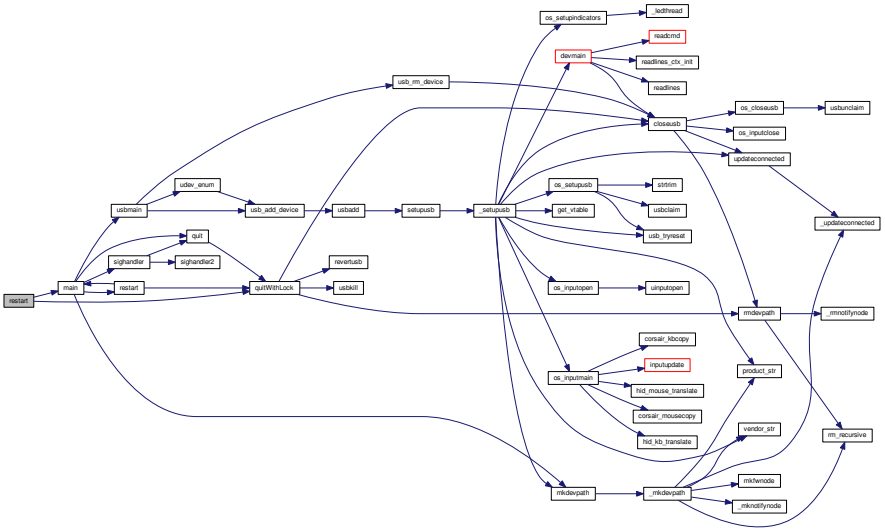
Definition at line 228 of file main.c.

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

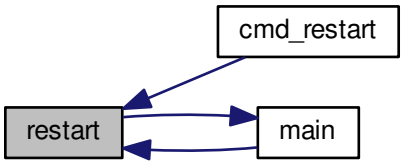
Referenced by cmd_restart(), and main().

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

Here is the call graph for this function:



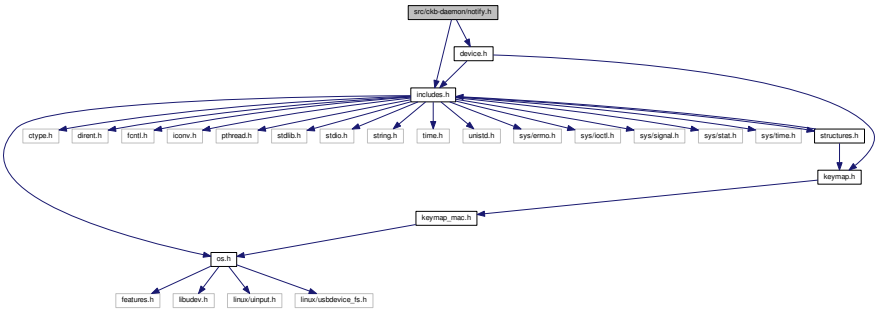
Here is the caller graph for this function:



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

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

Include dependency graph for notify.h:



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

Functions

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

9.34.1 Function Documentation

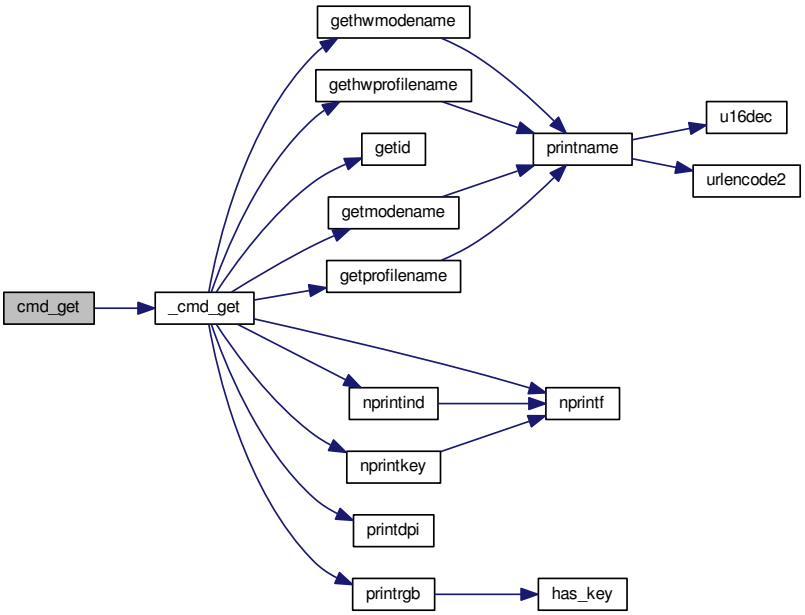
9.34.1.1 void cmd_get (usbdevice ∗ *kb,* usbmode ∗ *mode,* int *nnumber,* int *dummy,* const char ∗ *setting*)

Definition at line 215 of file notify.c.

References _cmd_get(), and imutex.

```
215                                                                                 {
216     (void)dummy;
217
218     pthread_mutex_lock(imutex(kb));
219     _cmd_get(kb, mode, nnumber, setting);
220     pthread_mutex_unlock(imutex(kb));
221 }
```

Here is the call graph for this function:



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

Definition at line 61 of file notify.c.

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

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

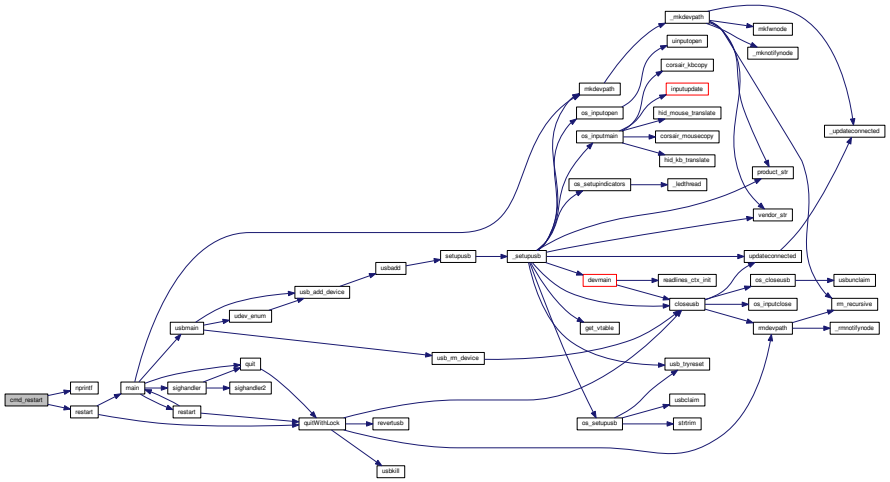
9.34.1.3 void cmd_restart (usbdevice * kb, usbmode * mode, int nnumber, int dummy, const char * content)

Definition at line 225 of file notify.c.

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

```
225                                                                                                 {
226     (void)mode;
227     (void)nnumber;
228     (void)dummy;
229
230     ckb_info("RESTART called with %s\n", content);
231     nprintf(kb, -1, 0, "RESTART called with %s\n", content);
232     restart();
233 }
```

Here is the call graph for this function:



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

Definition at line 8 of file notify.c.

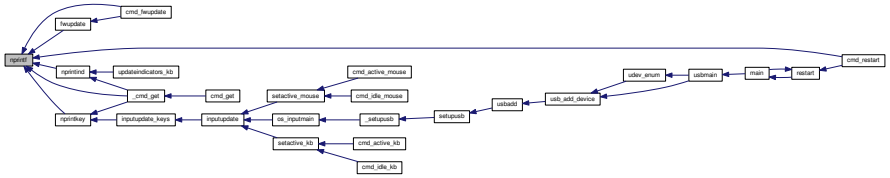
References INDEX_OF, usbprofile::mode, usbdevice::outfifo, OUTFIFO_MAX, and usbdevice::profile.

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

```
8                                                                                                   {
9     if(!kb)
```

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

Here is the caller graph for this function:



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

Definition at line 43 of file notify.c.

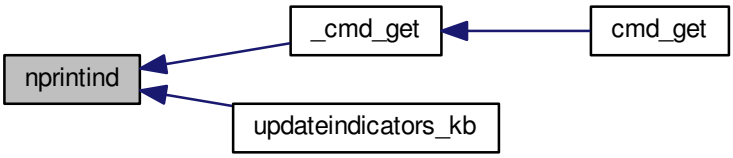
References I_CAPS, I_NUM, I_SCROLL, and nprintf().

Referenced by _cmd_get(), and updateindicators_kb().

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

Here is the call graph for this function:

Here is the caller graph for this function:



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

Definition at line 35 of file notify.c.

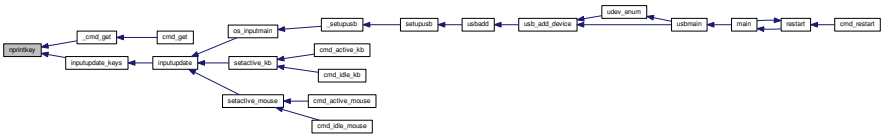
References keymap, key::name, and nprintf().

Referenced by _cmd_get(), and inputupdate_keys().

```
35                                                                  {
36     const key* map = keymap + keyindex;
37     if(map->name)
38         nprintf(kb, nnumber, 0, "key %c%s\n", down ? '+' : '-', map->name);
39     else
40         nprintf(kb, nnumber, 0, "key %c#%d\n", down ? '+' : '-', keyindex);
41 }
```

Here is the call graph for this function:

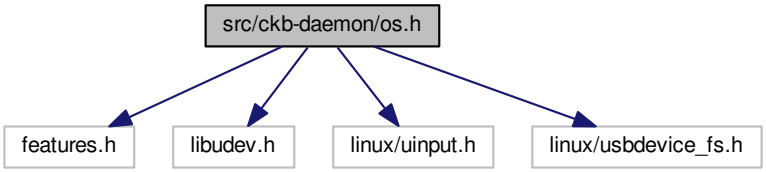
Here is the caller graph for this function:



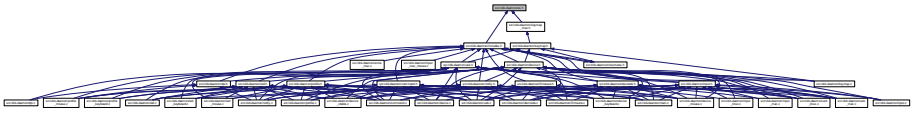
9.35 src/ckb-daemon/os.h File Reference

```
#include <features.h>
#include <libudev.h>
#include <linux/uinput.h>
#include <linux/usbdevice_fs.h>
```

Include dependency graph for os.h:



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



Macros

- #define _DEFAULT_SOURCE
- #define _GNU_SOURCE
- #define UINPUT_VERSION 2
- #define euid_guard_start
- #define euid_guard_stop

9.35.1 Macro Definition Documentation

9.35.1.1 #define _DEFAULT_SOURCE

Definition at line 22 of file os.h.

9.35.1.2 #define _GNU_SOURCE

Definition at line 26 of file os.h.

9.35.1.3 #define euid_guard_start

Definition at line 40 of file os.h.

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

9.35.1.4 #define euid_guard_stop

Definition at line 41 of file os.h.

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

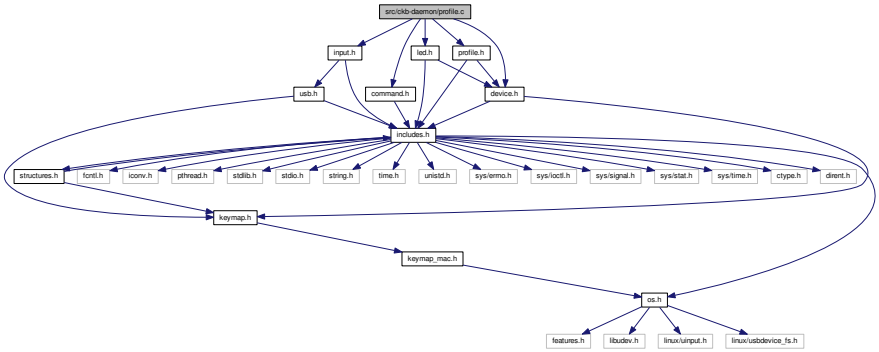
9.35.1.5 #define UINPUT_VERSION 2

Definition at line 35 of file os.h.

9.36 src/ckb-daemon/profile.c File Reference

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

Include dependency graph for profile.c:



Functions

- void urldecode2 (char *dst, const char *src)
- void urlencode2 (char *dst, const char *src)
- int setid (usbid *id, const char *guid)
- char * getid (usbid *id)
- void u16enc (char *in, ushort *out, size_t *srclen, size_t *dstlen)
- void u16dec (ushort *in, char *out, size_t *srclen, size_t *dstlen)
- void cmd_name (usbdevice *kb, usbmode *mode, int dummy1, int dummy2, const char *name)
- void cmd_profilename (usbdevice *kb, usbmode *dummy1, int dummy2, int dummy3, const char *name)

- char * printname (ushort *name, int length)
- char * getmodename (usbmode *mode)
- char * getprofilename (usbprofile *profile)
- char * gethwmodename (hwprofile *profile, int index)
- char * gethwprofilename (hwprofile *profile)
- void cmd_id (usbdevice *kb, usbmode *mode, int dummy1, int dummy2, const char *id)
- void cmd_profileid (usbdevice *kb, usbmode *mode, int dummy1, int dummy2, const char *id)
- static void initmode (usbmode *mode)
- void allocprofile (usbdevice *kb)
- int loadprofile (usbdevice *kb)
- static void freemode (usbmode *mode)
- void cmd_erase (usbdevice *kb, usbmode *mode, int dummy1, int dummy2, const char *dummy3)
- static void _freeprofile (usbdevice *kb)
- void cmd_eraseprofile (usbdevice *kb, usbmode *dummy1, int dummy2, int dummy3, const char *dummy4)
- void freeprofile (usbdevice *kb)
- void hwtonative (usbprofile *profile, hwprofile *hw, int modecount)
- void nativetohw (usbprofile *profile, hwprofile *hw, int modecount)

Variables

- static iconv_t utf8to16 = 0
- static iconv_t utf16to8 = 0

9.36.1 Function Documentation

9.36.1.1 static void _freeprofile (usbdevice * *kb*) [static]

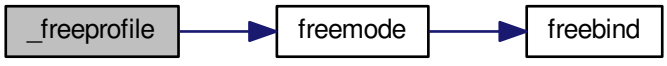
Definition at line 230 of file profile.c.

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

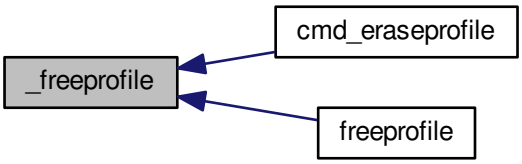
Referenced by cmd_eraseprofile(), and freeprofile().

```
230                                         {
231     usbprofile* profile = kb->profile;
232     if(!profile)
233         return;
234     // Clear all mode data
235     for(int i = 0; i < MODE_COUNT; i++)
236         freemode(profile->mode + i);
237     free(profile);
238     kb->profile = 0;
239 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



9.36.1.2 void allocprofile (usbdevice * kb)

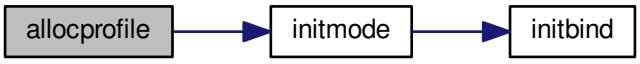
Definition at line 198 of file profile.c.

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

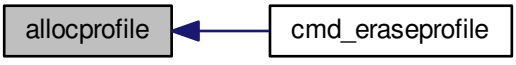
Referenced by cmd_eraseprofile().

```
198                                {
199     if(kb->profile)
200         return;
201     usbprofile* profile = kb->profile = calloc(1, sizeof(
      usbprofile));
202     for(int i = 0; i < MODE_COUNT; i++)
203         initmode(profile->mode + i);
204     profile->currentmode = profile->mode;
205     profile->lastlight.forceupdate = profile->lastdpi.
      forceupdate = 1;
206 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



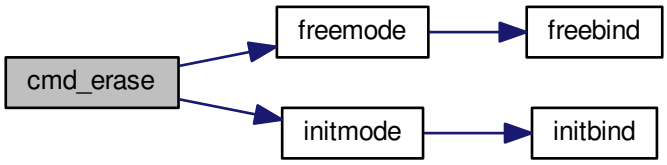
9.36.1.3 void cmd_erase (usbdevice * *kb*, usbmode * *mode*, int *dummy1*, int *dummy2*, const char * *dummy3*)

Definition at line 219 of file profile.c.

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

```
219                                                                                 {
220     (void)dummy1;
221     (void)dummy2;
222     (void)dummy3;
223
224     pthread_mutex_lock(imutex(kb));
225     freemode(mode);
226     initmode(mode);
227     pthread_mutex_unlock(imutex(kb));
228 }
```

Here is the call graph for this function:



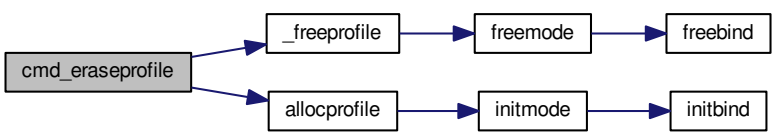
9.36.1.4 void cmd_eraseprofile (usbdevice * *kb*, usbmode * *dummy1*, int *dummy2*, int *dummy3*, const char * *dummy4*)

Definition at line 241 of file profile.c.

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

```
241                                                                                 {
242     (void)dummy1;
243     (void)dummy2;
244     (void)dummy3;
245     (void)dummy4;
246
247     pthread_mutex_lock(imutex(kb));
248     _freeprofile(kb);
249     allocprofile(kb);
250     pthread_mutex_unlock(imutex(kb));
251 }
```

Here is the call graph for this function:



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

Definition at line 168 of file profile.c.

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

```
168                                                                                 {
169     (void)kb;
170     (void)dummy1;
171     (void)dummy2;
172 
173     // ID is either a GUID or an 8-digit hex number
174     int newmodified;
175     if(!setid(&mode->id, id) && sscanf(id, "%08x", &newmodified) == 1)
176         memcpy(mode->id.modified, &newmodified, sizeof(newmodified));
177 }
```

Here is the call graph for this function:

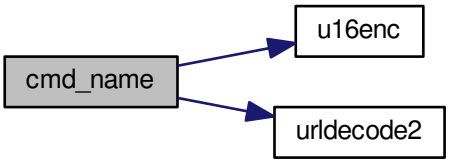
9.36.1.6 void cmd_name (usbdevice * kb, usbmode * mode, int dummy1, int dummy2, const char * name)

Definition at line 117 of file profile.c.

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

```
117                                                                                 {
118     (void)kb;
119     (void)dummy1;
120     (void)dummy2;
121 
122     char decoded[strlen(name) + 1];
123     urldecode2(decoded, name);
124     size_t srclen = strlen(decoded), dstlen = MD_NAME_LEN;
125     u16enc(decoded, mode->name, &srclen, &dstlen);
126 }
```

Here is the call graph for this function:



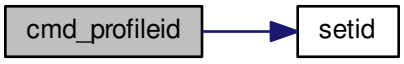
9.36.1.7 void cmd_profileid (usbdevice * *kb*, usbmode * *mode*, int *dummy1*, int *dummy2*, const char * *id*)

Definition at line 179 of file profile.c.

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

```
179                                                                                                 {
180     (void)mode;
181     (void)dummy1;
182     (void)dummy2;
183
184     usbprofile* profile = kb->profile;
185     int newmodified;
186     if(!setid(&profile->id, id) && sscanf(id, "%08x", &newmodified) == 1)
187         memcpy(profile->id.modified, &newmodified, sizeof(newmodified));
188
189 }
```

Here is the call graph for this function:



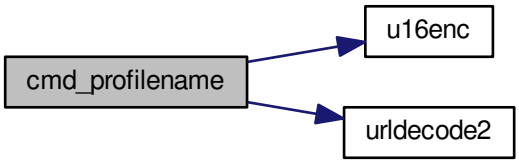
9.36.1.8 void cmd_profilename (usbdevice * *kb*, usbmode * *dummy1*, int *dummy2*, int *dummy3*, const char * *name*)

Definition at line 128 of file profile.c.

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

```
128                                                                                                 {
129     (void)dummy1;
130     (void)dummy2;
131     (void)dummy3;
132
133     usbprofile* profile = kb->profile;
134     char decoded[strlen(name) + 1];
135     urldecode2(decoded, name);
136     size_t srclen = strlen(decoded), dstlen = PR_NAME_LEN;
137     u16enc(decoded, profile->name, &srclen, &dstlen);
138 }
```

Here is the call graph for this function:



9.36.1.9 static void freemode (usbmode * *mode*) [static]

Definition at line 214 of file profile.c.

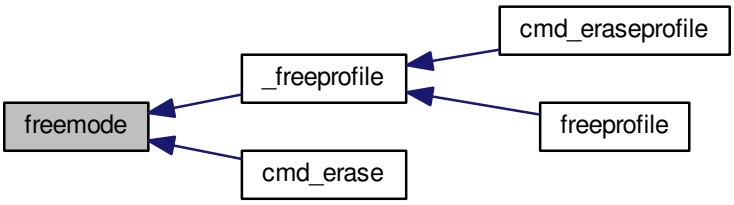
References usbmode::bind, and freebind().

Referenced by _freeprofile(), and cmd_erase().

```
214                                 {
215     freebind(&mode->bind);
216     memset(mode, 0, sizeof(*mode));
217 }
```

Here is the call graph for this function:

Here is the caller graph for this function:



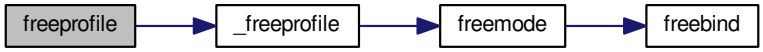
9.36.1.10 void freeprofile (usbdevice * *kb*)

Definition at line 253 of file profile.c.

References _freeprofile(), and usbdevice::hw.

```
253                                 {
254     _freeprofile(kb);
255     // Also free HW profile
256     free(kb->hw);
257     kb->hw = 0;
258 }
```

Here is the call graph for this function:



9.36.1.11 char* gethwmodename (hwprofile * *profile*, int *index*)

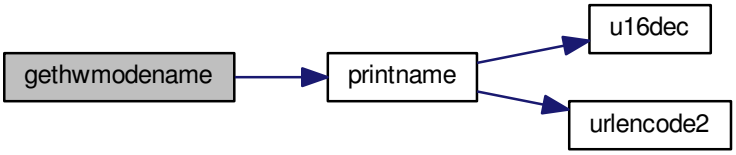
Definition at line 160 of file profile.c.

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

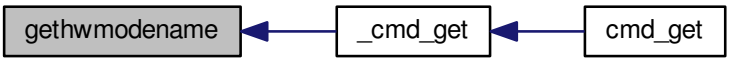
Referenced by _cmd_get().

```
160                                                 {
161     return printname(profile->name[index + 1], MD_NAME_LEN);
162 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



9.36.1.12 char* gethwprofilename (hwprofile * *profile*)

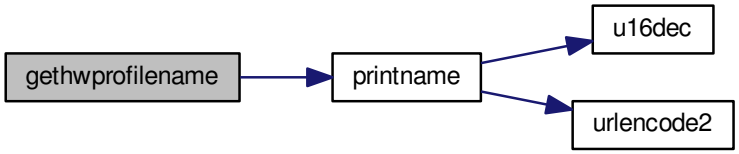
Definition at line 164 of file profile.c.

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

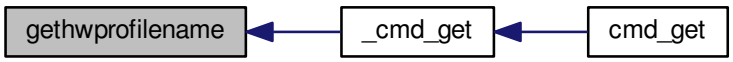
Referenced by _cmd_get().

```
164                                         {
165     return printname(profile->name[0], MD_NAME_LEN);
166 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



9.36.1.13 char* getid (usbid * *id*)

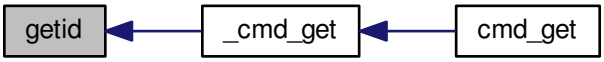
Definition at line 79 of file profile.c.

References usbid::guid.

Referenced by _cmd_get().

```
79                         {
80     int32_t data1;
81     int16_t data2, data3, data4a;
82     char data4b[6];
83     memcpy(&data1, id->guid + 0x0, 4);
84     memcpy(&data2, id->guid + 0x4, 2);
85     memcpy(&data3, id->guid + 0x6, 2);
86     memcpy(&data4a, id->guid + 0x8, 2);
87     memcpy(data4b, id->guid + 0xA, 6);
88     char* guid = malloc(39);
89     snprintf(guid, 39, "{%08X-%04hX-%04hX-%04hX-%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX}",
90              data1, data2, data3, data4a, data4b[0], data4b[1], data4b[2], data4b[3], data4b[4], data4b[5])
     ;
91     return guid;
92 }
```

Here is the caller graph for this function:



9.36.1.14 char* getmodename (usbmode * *mode*)

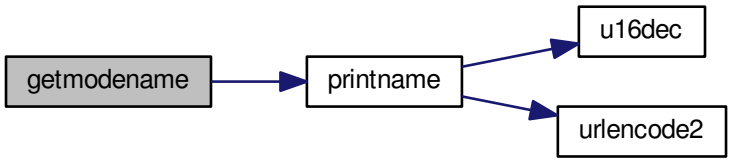
Definition at line 152 of file profile.c.

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

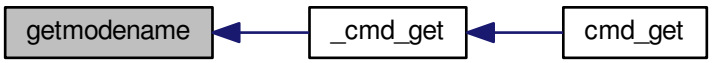
Referenced by _cmd_get().

```
152                                 {
153     return printname(mode->name, MD_NAME_LEN);
154 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



9.36.1.15 char* getprofilename (usbprofile * *profile*)

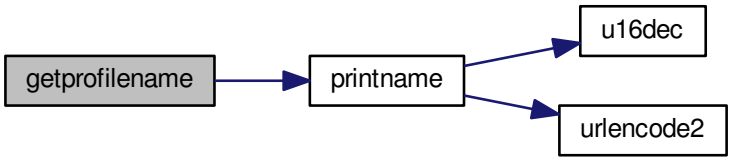
Definition at line 156 of file profile.c.

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

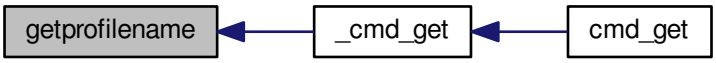
Referenced by _cmd_get().

```
156                                         {
157     return printname(profile->name, PR_NAME_LEN);
158 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



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

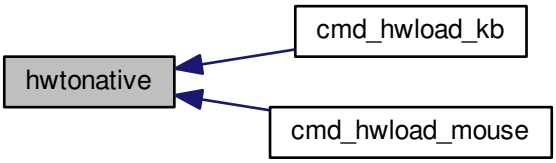
Definition at line 260 of file profile.c.

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().

```
260                                                                     {
261     // Copy the profile name and ID
262     memcpy(profile->name, hw->name[0], PR_NAME_LEN * 2);
263     memcpy(&profile->id, hw->id, sizeof(usbid));
264     // Copy the mode settings
265     for(int i = 0; i < modecount; i++){
266         usbmode* mode = profile->mode + i;
267         memcpy(mode->name, hw->name[i + 1], MD_NAME_LEN * 2);
268         memcpy(&mode->id, hw->id + i + 1, sizeof(usbid));
269         memcpy(&mode->light, hw->light + i, sizeof(lighting));
270         memcpy(&mode->dpi, hw->dpi + i, sizeof(dpiset));
271         // Set a force update on the light/DPI since they've been overwritten
272         mode->light.forceupdate = mode->dpi.forceupdate = 1;
273     }
274     profile->lastlight.forceupdate = profile->lastdpi.
    forceupdate = 1;
275 }
```

Here is the caller graph for this function:



9.36.1.17 static void initmode (usbmode * *mode*) [static]

Definition at line 191 of file profile.c.

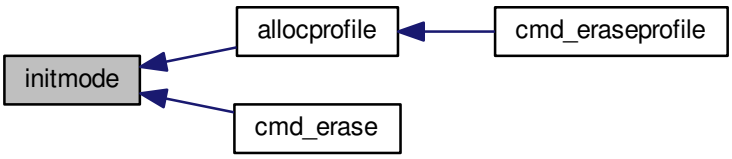
References usbmode::bind, usbmode::dpi, dpiset::forceupdate, lighting::forceupdate, initbind(), and usbmode::light.

Referenced by allocprofile(), and cmd_erase().

```
191                                     {
192     memset(mode, 0, sizeof(*mode));
193     mode->light.forceupdate = 1;
194     mode->dpi.forceupdate = 1;
195     initbind(&mode->bind);
196 }
```

Here is the call graph for this function:

Here is the caller graph for this function:



9.36.1.18 int loadprofile (usbdevice * kb)

Definition at line 208 of file profile.c.

References hwloadprofile.

```
208                                {
209     if(hwloadprofile(kb, 1))
210         return -1;
211     return 0;
212 }
```

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

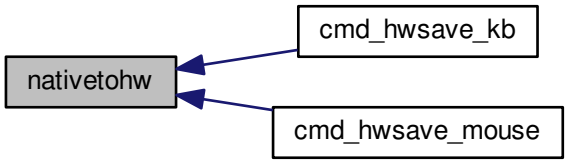
Definition at line 277 of file profile.c.

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().

```
277                                                                    {
278     // Copy name and ID
279     memcpy(hw->name[0], profile->name, PR_NAME_LEN * 2);
280     memcpy(hw->id, &profile->id, sizeof(usbid));
281     // Copy the mode settings
282     for(int i = 0; i < modecount; i++){
283         usbmode* mode = profile->mode + i;
284         memcpy(hw->name[i + 1], mode->name, MD_NAME_LEN * 2);
285         memcpy(hw->id + i + 1, &mode->id, sizeof(usbid));
286         memcpy(hw->light + i, &mode->light, sizeof(lighting));
287         memcpy(hw->dpi + i, &mode->dpi, sizeof(dpiset));
288     }
289 }
```

Here is the caller graph for this function:



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

Definition at line 140 of file profile.c.

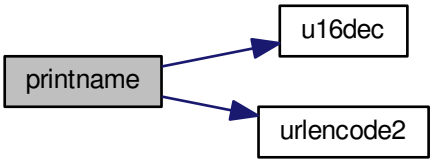
References u16dec(), and urlencode2().

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

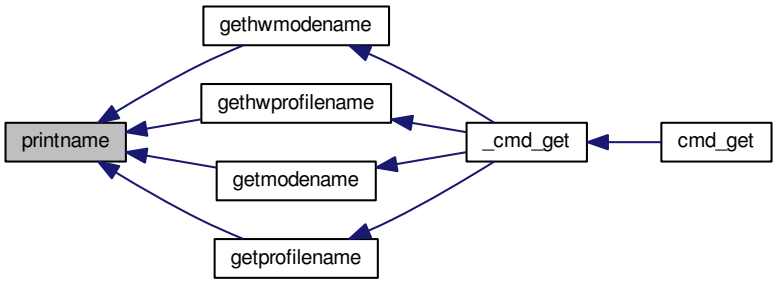
```
140                                        {
141     // Convert the name to UTF-8
142     char* buffer = calloc(1, length * 4 - 3);
143     size_t srclen = length, dstlen = length * 4 - 4;
144     u16dec(name, buffer, &srclen, &dstlen);
145     // URL-encode it
```

```
146     char* buffer2 = malloc(strlen(buffer) * 3 + 1);
147     urlencode2(buffer2, buffer);
148     free(buffer);
149     return buffer2;
150 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



9.36.1.21 int setid (usbid * *id*, const char * *guid*)

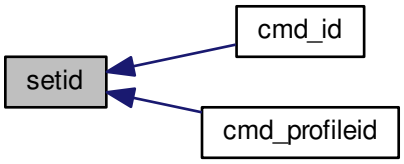
Definition at line 64 of file profile.c.

References usbid::guid.

Referenced by cmd_id(), and cmd_profileid().

```
64                                      {
65     int32_t data1;
66     int16_t data2, data3, data4a;
67     char data4b[6];
68     if(sscanf(guid, "{%08X-%04hX-%04hX-%04hX-%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX}",
69               &data1, &data2, &data3, &data4a, data4b, data4b + 1, data4b + 2, data4b + 3, data4b + 4,
      data4b + 5) != 10)
70         return 0;
71     memcpy(id->guid + 0x0, &data1, 4);
72     memcpy(id->guid + 0x4, &data2, 2);
73     memcpy(id->guid + 0x6, &data3, 2);
74     memcpy(id->guid + 0x8, &data4a, 2);
75     memcpy(id->guid + 0xA, data4b, 6);
76     return 1;
77 }
```

Here is the caller graph for this function:



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

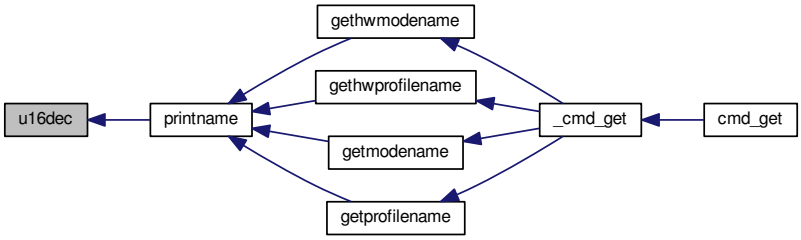
Definition at line 105 of file profile.c.

References utf16to8.

Referenced by printname().

```
105                                                                 {
106     if(!utf16to8)
107         utf16to8 = iconv_open("UTF-8", "UTF-16LE");
108     size_t srclen2 = 0, srclenmax = *srclen;
109     for(; srclen2 < srclenmax; srclen2++){
110         if(!in[srclen2])
111             break;
112     }
113     *srclen = srclen2 * 2;
114     iconv(utf16to8, (char**)&in, srclen, &out, dstlen);
115 }
```

Here is the caller graph for this function:



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

Definition at line 97 of file profile.c.

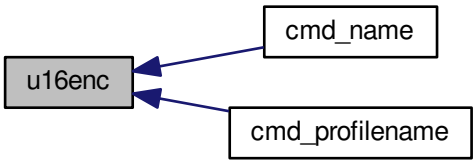
References utf8to16.

Referenced by cmd_name(), and cmd_profilename().

```
97                                                                  {
```

```
98      if(!utf8to16)
99          utf8to16 = iconv_open("UTF-16LE", "UTF-8");
100      memset(out, 0, *dstlen * 2);
101      *dstlen = *dstlen * 2 - 2;
102      iconv(utf8to16, &in, srclen, (char**)&out, dstlen);
103  }
```

Here is the caller graph for this function:



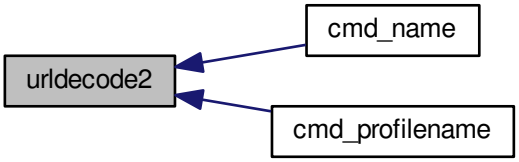
9.36.1.24 void urldecode2 (char * *dst*, const char * *src*)

Definition at line 8 of file profile.c.

Referenced by cmd_name(), and cmd_profilename().

```
8                                           {
9       char a, b;
10      char s;
11      while((s = *src)){
12          if((s == '%') &&
13                  ((a = src[1]) && (b = src[2])) &&
14                  (isxdigit(a) && isxdigit(b))){
15              if(a >= 'a')
16                  a -= 'a'-'A';
17              if(a >= 'A')
18                  a -= 'A' - 10;
19              else
20                  a -= '0';
21              if(b >= 'a')
22                  b -= 'a'-'A';
23              if(b >= 'A')
24                  b -= 'A' - 10;
25              else
26                  b -= '0';
27              *dst++ = 16 * a + b;
28              src += 3;
29          } else {
30              *dst++ = s;
31              src++;
32          }
33      }
34      *dst = '\0';
35  }
```

Here is the caller graph for this function:



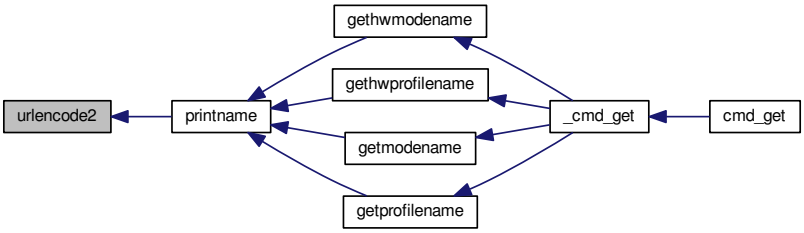
9.36.1.25 void urlencode2 (char * *dst*, const char * *src*)

Definition at line 37 of file profile.c.

Referenced by printname().

```
37                                              {
38     char s;
39     while((s = *src++)){
40         if(s <= ',' || s == '/' ||
41                 (s >= ':' && s <= '@') ||
42                 s == '[' || s == ']' ||
43                 s >= 0x7F){
44             char a = s >> 4, b = s & 0xF;
45             if(a >= 10)
46                 a += 'A' - 10;
47             else
48                 a += '0';
49             if(b >= 10)
50                 b += 'A' - 10;
51             else
52                 b += '0';
53             dst[0] = '%';
54             dst[1] = a;
55             dst[2] = b;
56             dst += 3;
57         } else
58             *dst++ = s;
59     }
60     *dst = '\0';
61 }
```

Here is the caller graph for this function:



9.36.2 Variable Documentation

9.36.2.1 iconv_t utf16to8 = 0 [static]

Definition at line 95 of file profile.c.

Referenced by u16dec().

9.36.2.2 iconv_t utf8to16 = 0 [static]

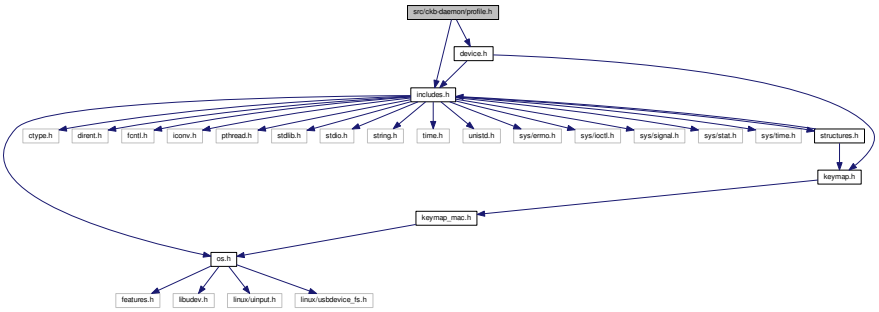
Definition at line 95 of file profile.c.

Referenced by u16enc().

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

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

Include dependency graph for profile.h:



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

Macros

- #define hwloadprofile(kb, apply) (kb)->vtable->hwload(kb, 0, 0, apply, 0)

Functions

- void allocprofile (usbdevice *kb)
- int loadprofile (usbdevice *kb)
- void freeprofile (usbdevice *kb)
- void cmd_erase (usbdevice *kb, usbmode *mode, int dummy1, int dummy2, const char *dummy3)
- void cmd_eraseprofile (usbdevice *kb, usbmode *dummy1, int dummy2, int dummy3, const char *dummy4)
- void cmd_name (usbdevice *kb, usbmode *mode, int dummy1, int dummy2, const char *name)
- void cmd_profilename (usbdevice *kb, usbmode *dummy1, int dummy2, int dummy3, const char *name)
- char * getmodename (usbmode *mode)
- char * getprofilename (usbprofile *profile)
- char * gethwmodename (hwprofile *profile, int index)

- char * gethwprofilename (hwprofile *profile)
- int setid (usbid *id, const char *guid)
- char * getid (usbid *id)
- void hwtonative (usbprofile *profile, hwprofile *hw, int modecount)
- void nativetohw (usbprofile *profile, hwprofile *hw, int modecount)
- void cmd_id (usbdevice *kb, usbmode *mode, int dummy1, int dummy2, const char *id)
- void cmd_profileid (usbdevice *kb, usbmode *mode, int dummy1, int dummy2, const char *id)
- int cmd_hwload_kb (usbdevice *kb, usbmode *dummy1, int dummy2, int apply, const char *dummy3)
- int cmd_hwload_mouse (usbdevice *kb, usbmode *dummy1, int dummy2, int apply, const char *dummy3)
- int cmd_hwsave_kb (usbdevice *kb, usbmode *dummy1, int dummy2, int dummy3, const char *dummy4)
- int cmd_hwsave_mouse (usbdevice *kb, usbmode *dummy1, int dummy2, int dummy3, const char *dummy4)

9.37.1 Macro Definition Documentation

9.37.1.1 #define hwloadprofile(*kb, apply*) (kb)->vtable->hwload(kb, 0, 0, apply, 0)

Definition at line 52 of file profile.h.

Referenced by _start_dev(), and loadprofile().

9.37.2 Function Documentation

9.37.2.1 void allocprofile (usbdevice * *kb*)

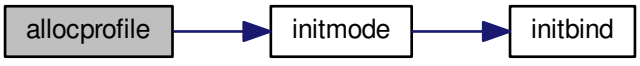
Definition at line 198 of file profile.c.

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

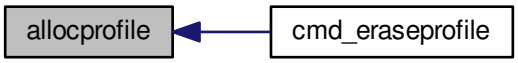
Referenced by cmd_eraseprofile().

```
198                                {
199     if(kb->profile)
200         return;
201     usbprofile* profile = kb->profile = calloc(1, sizeof(
      usbprofile));
202     for(int i = 0; i < MODE_COUNT; i++)
203         initmode(profile->mode + i);
204     profile->currentmode = profile->mode;
205     profile->lastlight.forceupdate = profile->lastdpi.
      forceupdate = 1;
206 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



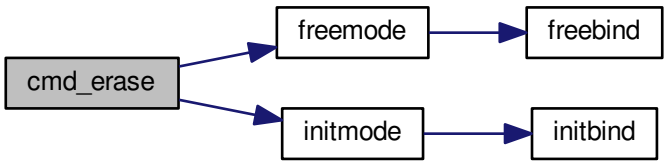
9.37.2.2 void cmd_erase (usbdevice * *kb*, usbmode * *mode*, int *dummy1*, int *dummy2*, const char * *dummy3*)

Definition at line 219 of file profile.c.

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

```
219                                                                                 {
220     (void)dummy1;
221     (void)dummy2;
222     (void)dummy3;
223
224     pthread_mutex_lock(imutex(kb));
225     freemode(mode);
226     initmode(mode);
227     pthread_mutex_unlock(imutex(kb));
228 }
```

Here is the call graph for this function:



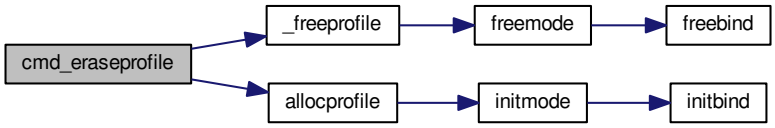
9.37.2.3 void cmd_eraseprofile (usbdevice * *kb*, usbmode * *dummy1*, int *dummy2*, int *dummy3*, const char * *dummy4*)

Definition at line 241 of file profile.c.

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

```
241                                                                                 {
242     (void)dummy1;
243     (void)dummy2;
244     (void)dummy3;
245     (void)dummy4;
246
247     pthread_mutex_lock(imutex(kb));
248     _freeprofile(kb);
249     allocprofile(kb);
250     pthread_mutex_unlock(imutex(kb));
251 }
```

Here is the call graph for this function:



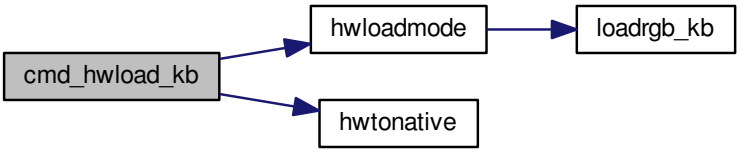
9.37.2.4 int cmd_hwload_kb (usbdevice * *kb*, usbmode * *dummy1*, int *dummy2*, int *apply*, const char * *dummy3*)

Definition at line 16 of file profile_keyboard.c.

References DELAY_LONG, usbdevice::hw, hwloadmode(), HWMODE_K70, HWMODE_K95, hwtonative(), hwprofile::id, IS_K95, MSG_SIZE, hwprofile::name, PR_NAME_LEN, usbdevice::profile, and usbrecv.

```
16                                                                                                  {
17      (void)dummy1;
18      (void)dummy2;
19      (void)dummy3;
20
21      DELAY_LONG(kb);
22      hwprofile* hw = calloc(1, sizeof(hwprofile));
23      // Ask for profile and mode IDs
24      uchar data_pkt[2][MSG_SIZE] = {
25          { 0x0e, 0x15, 0x01, 0 },
26          { 0x0e, 0x16, 0x01, 0 }
27      };
28      uchar in_pkt[MSG_SIZE];
29      int modes = (IS_K95(kb) ? HWMODE_K95 : HWMODE_K70);
30      for(int i = 0; i <= modes; i++){
31          data_pkt[0][3] = i;
32          if(!usbrecv(kb, data_pkt[0], in_pkt)){
33              free(hw);
34              return -1;
35          }
36          memcpy(hw->id + i, in_pkt + 4, sizeof(usbid));
37      }
38      // Ask for profile name
39      if(!usbrecv(kb, data_pkt[1], in_pkt)){
40          free(hw);
41          return -1;
42      }
43      memcpy(hw->name[0], in_pkt + 4, PR_NAME_LEN * 2);
44      // Load modes
45      for(int i = 0; i < modes; i++){
46          if(hwloadmode(kb, hw, i)){
47              free(hw);
48              return -1;
49          }
50      }
51      // Make the profile active (if requested)
52      if(apply)
53          hwtonative(kb->profile, hw, modes);
54      // Free the existing profile (if any)
55      free(kb->hw);
56      kb->hw = hw;
57      DELAY_LONG(kb);
58      return 0;
59  }
```

Here is the call graph for this function:



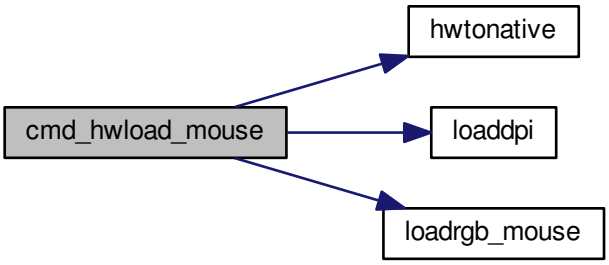
9.37.2.5 int cmd_hwload_mouse (usbdevice * *kb*, usbmode * *dummy1*, int *dummy2*, int *apply*, const char * *dummy3*)

Definition at line 6 of file profile_mouse.c.

References DELAY_LONG, hwprofile::dpi, usbdevice::hw, hwtonative(), hwprofile::id, hwprofile::light, loaddpi(), loadrgb_mouse(), MSG_SIZE, hwprofile::name, PR_NAME_LEN, usbdevice::profile, and usbrecv.

```
6                                                                                                   {
7       (void)dummy1;
8       (void)dummy2;
9       (void)dummy3;
10
11      DELAY_LONG(kb);
12      hwprofile* hw = calloc(1, sizeof(hwprofile));
13      // Ask for profile and mode IDs
14      uchar data_pkt[2][MSG_SIZE] = {
15          { 0x0e, 0x15, 0x01, 0 },
16          { 0x0e, 0x16, 0x01, 0 }
17      };
18      uchar in_pkt[MSG_SIZE];
19      for(int i = 0; i <= 1; i++){
20          data_pkt[0][3] = i;
21          if(!usbrecv(kb, data_pkt[0], in_pkt)){
22              free(hw);
23              return -1;
24          }
25          memcpy(hw->id + i, in_pkt + 4, sizeof(usbid));
26      }
27      // Ask for profile and mode names
28      for(int i = 0; i <= 1; i++){
29          data_pkt[1][3] = i;
30          if(!usbrecv(kb, data_pkt[1],in_pkt)){
31              free(hw);
32              return -1;
33          }
34          memcpy(hw->name[i], in_pkt + 4, PR_NAME_LEN * 2);
35      }
36
37      // Load the RGB and DPI settings
38      if(loadrgb_mouse(kb, hw->light, 0)
39              || loaddpi(kb, hw->dpi, hw->light)){
40          free(hw);
41          return -1;
42      }
43
44      // Make the profile active (if requested)
45      if(apply)
46          hwtonative(kb->profile, hw, 1);
47      // Free the existing profile (if any)
48      free(kb->hw);
49      kb->hw = hw;
50      DELAY_LONG(kb);
51      return 0;
52  }
```

Here is the call graph for this function:



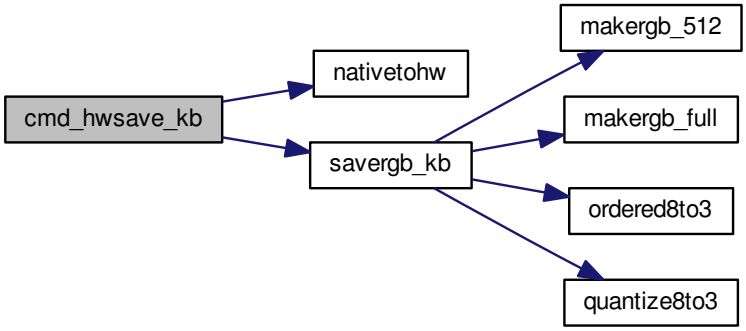
9.37.2.6 int cmd_hwsave_kb (usbdevice * *kb,* usbmode * *dummy1,* int *dummy2,* int *dummy3,* const char * *dummy4*)

Definition at line 61 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.

```
61                                                                                  {
62      (void)dummy1;
63      (void)dummy2;
64      (void)dummy3;
65      (void)dummy4;
66
67      DELAY_LONG(kb);
68      hwprofile* hw = kb->hw;
69      if(!hw)
70          hw = kb->hw = calloc(1, sizeof(hwprofile));
71      int modes = (IS_K95(kb) ? HWMODE_K95 : HWMODE_K70);
72      nativetohw(kb->profile, hw, modes);
73      // Save the profile and mode names
74      uchar data_pkt[2][MSG_SIZE] = {
75          { 0x07, 0x16, 0x01, 0 },
76          { 0x07, 0x15, 0x01, 0 },
77      };
78      // Save the mode names
79      for(int i = 0; i <= modes; i++){
80          data_pkt[0][3] = i;
81          memcpy(data_pkt[0] + 4, hw->name[i], MD_NAME_LEN * 2);
82          if(!usbsend(kb, data_pkt[0], 1))
83              return -1;
84      }
85      // Save the IDs
86      for(int i = 0; i <= modes; i++){
87          data_pkt[1][3] = i;
88          memcpy(data_pkt[1] + 4, hw->id + i, sizeof(usbid));
89          if(!usbsend(kb, data_pkt[1], 1))
90              return -1;
91      }
92      // Save the RGB data
93      for(int i = 0; i < modes; i++){
94          if(savergb_kb(kb, hw->light + i, i))
95              return -1;
96      }
97      DELAY_LONG(kb);
98      return 0;
99  }
```

Here is the call graph for this function:



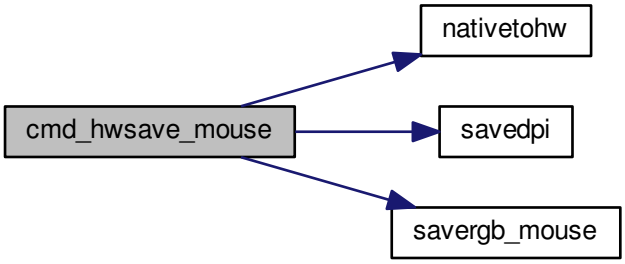
9.37.2.7 int cmd_hwsave_mouse (usbdevice * *kb*, usbmode * *dummy1*, int *dummy2*, int *dummy3*, const char * *dummy4*)

Definition at line 54 of file profile_mouse.c.

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

```
54                                                                                                  {
55      (void)dummy1;
56      (void)dummy2;
57      (void)dummy3;
58      (void)dummy4;
59
60      DELAY_LONG(kb);
61      hwprofile* hw = kb->hw;
62      if(!hw)
63          hw = kb->hw = calloc(1, sizeof(hwprofile));
64      nativetohw(kb->profile, hw, 1);
65      // Save the profile and mode names
66      uchar data_pkt[2][MSG_SIZE] = {
67          { 0x07, 0x16, 0x01, 0 },
68          { 0x07, 0x15, 0x01, 0 },
69      };
70      for(int i = 0; i <= 1; i++){
71          data_pkt[0][3] = i;
72          memcpy(data_pkt[0] + 4, hw->name[i], MD_NAME_LEN * 2);
73          if(!usbsend(kb, data_pkt[0], 1))
74              return -1;
75      }
76      // Save the IDs
77      for(int i = 0; i <= 1; i++){
78          data_pkt[1][3] = i;
79          memcpy(data_pkt[1] + 4, hw->id + i, sizeof(usbid));
80          if(!usbsend(kb, data_pkt[1], 1))
81              return -1;
82      }
83      // Save the RGB data for the non-DPI zones
84      if(savergb_mouse(kb, hw->light, 0))
85          return -1;
86      // Save the DPI data (also saves RGB for those states)
87      if(savedpi(kb, hw->dpi, hw->light))
88          return -1;
89      DELAY_LONG(kb);
90      return 0;
91  }
```

Here is the call graph for this function:



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

Definition at line 168 of file profile.c.

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

```
168                                                                                 {
169     (void)kb;
170     (void)dummy1;
171     (void)dummy2;
172
173     // ID is either a GUID or an 8-digit hex number
174     int newmodified;
175     if(!setid(&mode->id, id) && sscanf(id, "%08x", &newmodified) == 1)
176         memcpy(mode->id.modified, &newmodified, sizeof(newmodified));
177 }
```

Here is the call graph for this function:

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

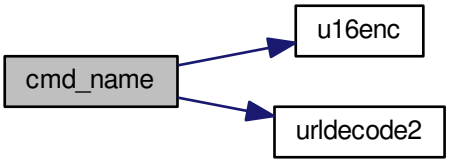
Definition at line 117 of file profile.c.

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

```
117                                                                                 {
118     (void)kb;
119     (void)dummy1;
120     (void)dummy2;
121
122     char decoded[strlen(name) + 1];
123     urldecode2(decoded, name);
```

```
124     size_t srclen = strlen(decoded), dstlen = MD_NAME_LEN;
125     u16enc(decoded, mode->name, &srclen, &dstlen);
126 }
```

Here is the call graph for this function:



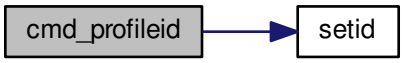
9.37.2.10 void cmd_profileid (usbdevice * kb, usbmode * mode, int dummy1, int dummy2, const char * id)

Definition at line 179 of file profile.c.

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

```
179                                                                                 {
180     (void)mode;
181     (void)dummy1;
182     (void)dummy2;
183
184     usbprofile* profile = kb->profile;
185     int newmodified;
186     if(!setid(&profile->id, id) && sscanf(id, "%08x", &newmodified) == 1)
187         memcpy(profile->id.modified, &newmodified, sizeof(newmodified));
188
189 }
```

Here is the call graph for this function:



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

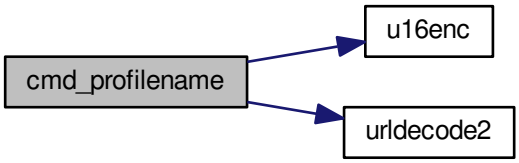
Definition at line 128 of file profile.c.

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

```
128                                                                                 {
129     (void)dummy1;
130     (void)dummy2;
131     (void)dummy3;
132
```

```
133     usbprofile* profile = kb->profile;
134     char decoded[strlen(name) + 1];
135     urldecode2(decoded, name);
136     size_t srclen = strlen(decoded), dstlen = PR_NAME_LEN;
137     u16enc(decoded, profile->name, &srclen, &dstlen);
138 }
```

Here is the call graph for this function:



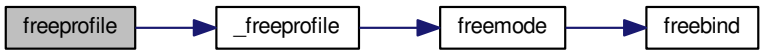
9.37.2.12 void freeprofile (usbdevice * *kb*)

Definition at line 253 of file profile.c.

References _freeprofile(), and usbdevice::hw.

```
253                                 {
254     _freeprofile(kb);
255     // Also free HW profile
256     free(kb->hw);
257     kb->hw = 0;
258 }
```

Here is the call graph for this function:



9.37.2.13 char* gethwmodename (hwprofile * *profile,* int *index*)

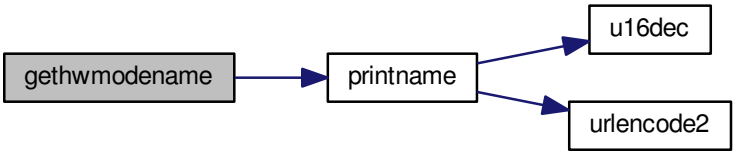
Definition at line 160 of file profile.c.

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

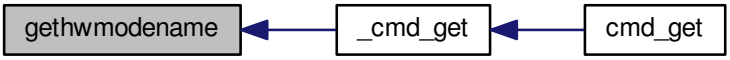
Referenced by _cmd_get().

```
160                                                 {
161     return printname(profile->name[index + 1], MD_NAME_LEN);
162 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



9.37.2.14 char* gethwprofilename (hwprofile * *profile*)

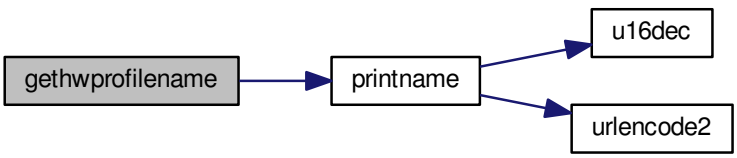
Definition at line 164 of file profile.c.

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

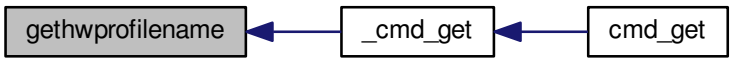
Referenced by _cmd_get().

```
164                                         {
165     return printname(profile->name[0], MD_NAME_LEN);
166 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



9.37.2.15 char* getid (usbid * *id*)

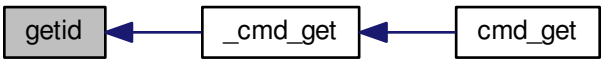
Definition at line 79 of file profile.c.

References usbid::guid.

Referenced by _cmd_get().

```
79                      {
80     int32_t data1;
81     int16_t data2, data3, data4a;
82     char data4b[6];
83     memcpy(&data1, id->guid + 0x0, 4);
84     memcpy(&data2, id->guid + 0x4, 2);
85     memcpy(&data3, id->guid + 0x6, 2);
86     memcpy(&data4a, id->guid + 0x8, 2);
87     memcpy(data4b, id->guid + 0xA, 6);
88     char* guid = malloc(39);
89     snprintf(guid, 39, "{%08X-%04hX-%04hX-%04hX-%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX}",
90              data1, data2, data3, data4a, data4b[0], data4b[1], data4b[2], data4b[3], data4b[4], data4b[5])
     ;
91     return guid;
92 }
```

Here is the caller graph for this function:



9.37.2.16 char* getmodename (usbmode * *mode*)

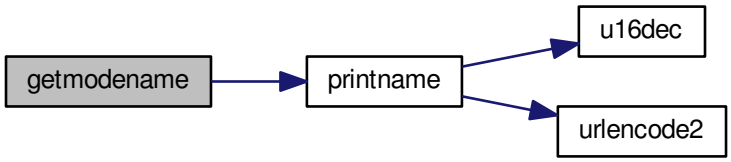
Definition at line 152 of file profile.c.

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

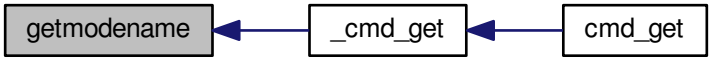
Referenced by _cmd_get().

```
152                                 {
153     return printname(mode->name, MD_NAME_LEN);
154 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



9.37.2.17 char* getprofilename (usbprofile * *profile*)

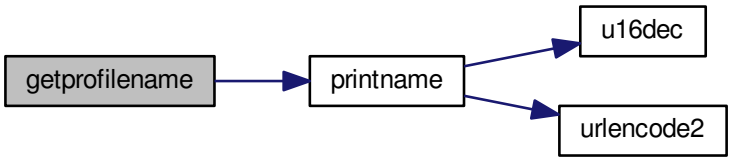
Definition at line 156 of file profile.c.

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

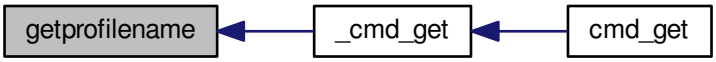
Referenced by _cmd_get().

```
156                                          {
157     return printname(profile->name, PR_NAME_LEN);
158 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



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

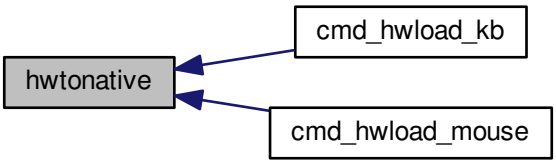
Definition at line 260 of file profile.c.

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().

```
260                                                                    {
261     // Copy the profile name and ID
262     memcpy(profile->name, hw->name[0], PR_NAME_LEN * 2);
263     memcpy(&profile->id, hw->id, sizeof(usbid));
264     // Copy the mode settings
265     for(int i = 0; i < modecount; i++){
266         usbmode* mode = profile->mode + i;
267         memcpy(mode->name, hw->name[i + 1], MD_NAME_LEN * 2);
268         memcpy(&mode->id, hw->id + i + 1, sizeof(usbid));
269         memcpy(&mode->light, hw->light + i, sizeof(lighting));
270         memcpy(&mode->dpi, hw->dpi + i, sizeof(dpiset));
271         // Set a force update on the light/DPI since they've been overwritten
272         mode->light.forceupdate = mode->dpi.forceupdate = 1;
273     }
274     profile->lastlight.forceupdate = profile->lastdpi.
      forceupdate = 1;
275 }
```

Here is the caller graph for this function:



9.37.2.19 int loadprofile (usbdevice * *kb*)

Definition at line 208 of file profile.c.

References hwloadprofile.

```
208                                 {
209     if(hwloadprofile(kb, 1))
210         return -1;
211     return 0;
212 }
```

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

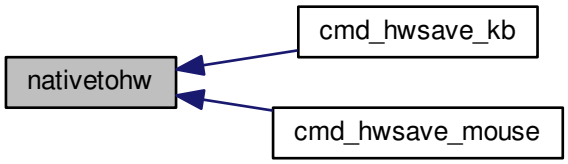
Definition at line 277 of file profile.c.

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().

```
277                                                                 {
278     // Copy name and ID
279     memcpy(hw->name[0], profile->name, PR_NAME_LEN * 2);
280     memcpy(hw->id, &profile->id, sizeof(usbid));
281     // Copy the mode settings
282     for(int i = 0; i < modecount; i++){
283         usbmode* mode = profile->mode + i;
284         memcpy(hw->name[i + 1], mode->name, MD_NAME_LEN * 2);
285         memcpy(hw->id + i + 1, &mode->id, sizeof(usbid));
286         memcpy(hw->light + i, &mode->light, sizeof(lighting));
287         memcpy(hw->dpi + i, &mode->dpi, sizeof(dpiset));
288     }
289 }
```

Here is the caller graph for this function:



9.37.2.21 int setid (usbid * *id,* const char * *guid*)

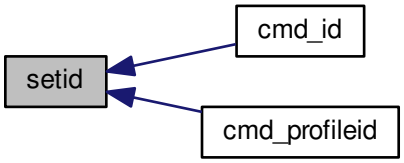
Definition at line 64 of file profile.c.

References usbid::guid.

Referenced by cmd_id(), and cmd_profileid().

```
64                                      {
65     int32_t data1;
66     int16_t data2, data3, data4a;
67     char data4b[6];
68     if(sscanf(guid, "{%08X-%04hX-%04hX-%04hX-%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX}",
69               &data1, &data2, &data3, &data4a, data4b, data4b + 1, data4b + 2, data4b + 3, data4b + 4,
      data4b + 5) != 10)
70         return 0;
71     memcpy(id->guid + 0x0, &data1, 4);
72     memcpy(id->guid + 0x4, &data2, 2);
73     memcpy(id->guid + 0x6, &data3, 2);
74     memcpy(id->guid + 0x8, &data4a, 2);
75     memcpy(id->guid + 0xA, data4b, 6);
76     return 1;
77 }
```

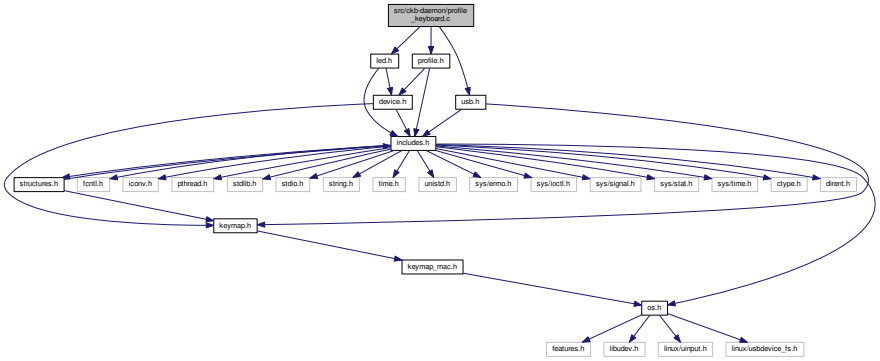
Here is the caller graph for this function:



9.38 src/ckb-daemon/profile_keyboard.c File Reference

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

Include dependency graph for profile_keyboard.c:



Functions

- static int hwloadmode (usbdevice ∗kb, hwprofile ∗hw, int mode)
- int cmd_hwload_kb (usbdevice ∗kb, usbmode ∗dummy1, int dummy2, int apply, const char ∗dummy3)
- int cmd_hwsave_kb (usbdevice ∗kb, usbmode ∗dummy1, int dummy2, int dummy3, const char ∗dummy4)

9.38.1 Function Documentation

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

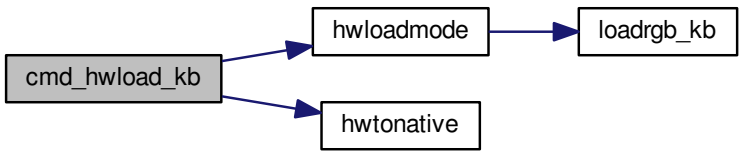
Definition at line 16 of file profile_keyboard.c.

References DELAY_LONG, usbdevice::hw, hwloadmode(), HWMODE_K70, HWMODE_K95, hwtonative(), hwprofile::id, IS_K95, MSG_SIZE, hwprofile::name, PR_NAME_LEN, usbdevice::profile, and usbrecv.

```
16                                                                      {
17     (void)dummy1;
18     (void)dummy2;
19     (void)dummy3;
```

```
20
21      DELAY_LONG(kb);
22      hwprofile* hw = calloc(1, sizeof(hwprofile));
23      // Ask for profile and mode IDs
24      uchar data_pkt[2][MSG_SIZE] = {
25          { 0x0e, 0x15, 0x01, 0 },
26          { 0x0e, 0x16, 0x01, 0 }
27      };
28      uchar in_pkt[MSG_SIZE];
29      int modes = (IS_K95(kb) ? HWMODE_K95 : HWMODE_K70);
30      for(int i = 0; i <= modes; i++){
31          data_pkt[0][3] = i;
32          if(!usbrecv(kb, data_pkt[0], in_pkt)){
33              free(hw);
34              return -1;
35          }
36          memcpy(hw->id + i, in_pkt + 4, sizeof(usbid));
37      }
38      // Ask for profile name
39      if(!usbrecv(kb, data_pkt[1], in_pkt)){
40          free(hw);
41          return -1;
42      }
43      memcpy(hw->name[0], in_pkt + 4, PR_NAME_LEN * 2);
44      // Load modes
45      for(int i = 0; i < modes; i++){
46          if(hwloadmode(kb, hw, i)){
47              free(hw);
48              return -1;
49          }
50      }
51      // Make the profile active (if requested)
52      if(apply)
53          hwtonative(kb->profile, hw, modes);
54      // Free the existing profile (if any)
55      free(kb->hw);
56      kb->hw = hw;
57      DELAY_LONG(kb);
58      return 0;
59  }
```

Here is the call graph for this function:



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

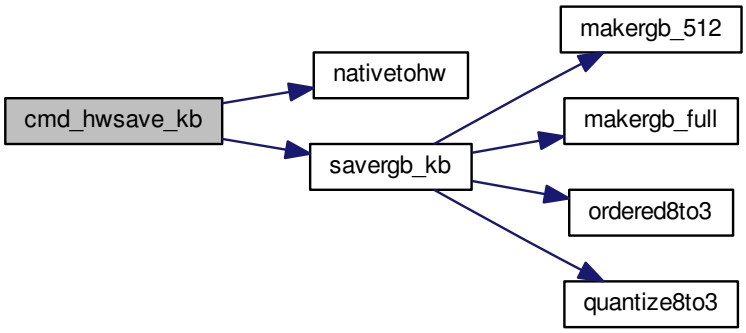
Definition at line 61 of file profile_keyboard.c.

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.

```
61                                                                                  {
62      (void)dummy1;
63      (void)dummy2;
64      (void)dummy3;
65      (void)dummy4;
66
67      DELAY_LONG(kb);
68      hwprofile* hw = kb->hw;
69      if(!hw)
70          hw = kb->hw = calloc(1, sizeof(hwprofile));
71      int modes = (IS_K95(kb) ? HWMODE_K95 : HWMODE_K70);
```

```
72      nativetohw(kb->profile, hw, modes);
73      // Save the profile and mode names
74      uchar data_pkt[2][MSG_SIZE] = {
75          { 0x07, 0x16, 0x01, 0 },
76          { 0x07, 0x15, 0x01, 0 },
77      };
78      // Save the mode names
79      for(int i = 0; i <= modes; i++){
80          data_pkt[0][3] = i;
81          memcpy(data_pkt[0] + 4, hw->name[i], MD_NAME_LEN * 2);
82          if(!usbsend(kb, data_pkt[0], 1))
83              return -1;
84      }
85      // Save the IDs
86      for(int i = 0; i <= modes; i++){
87          data_pkt[1][3] = i;
88          memcpy(data_pkt[1] + 4, hw->id + i, sizeof(usbid));
89          if(!usbsend(kb, data_pkt[1], 1))
90              return -1;
91      }
92      // Save the RGB data
93      for(int i = 0; i < modes; i++){
94          if(savergb_kb(kb, hw->light + i, i))
95              return -1;
96      }
97      DELAY_LONG(kb);
98      return 0;
99  }
```

Here is the call graph for this function:



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

Definition at line 5 of file profile_keyboard.c.

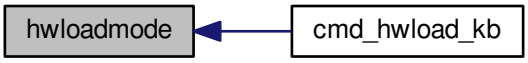
References hwprofile::light, loadrgb_kb(), MD_NAME_LEN, MSG_SIZE, hwprofile::name, and usbrecv.

Referenced by cmd_hwload_kb().

```
5                                                       {
6       // Ask for mode's name
7       uchar data_pkt[MSG_SIZE] = { 0x0e, 0x16, 0x01, mode + 1, 0 };
8       uchar in_pkt[MSG_SIZE];
9       if(!usbrecv(kb, data_pkt, in_pkt))
10          return -1;
11      memcpy(hw->name[mode + 1], in_pkt + 4, MD_NAME_LEN * 2);
12      // Load the RGB setting
13      return loadrgb_kb(kb, hw->light + mode, mode);
14  }
```

Here is the call graph for this function:

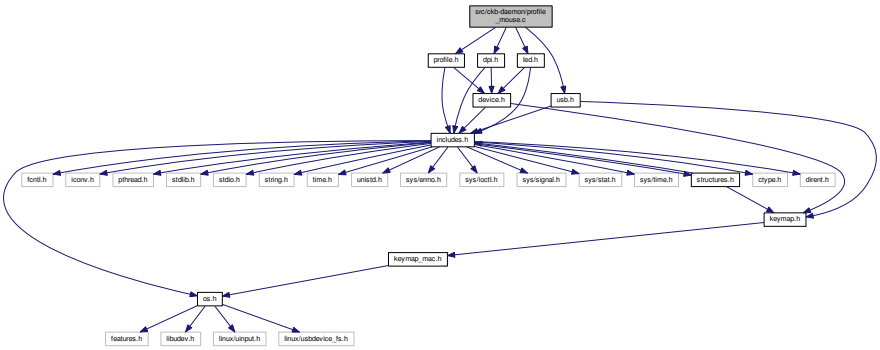
Here is the caller graph for this function:



9.39 src/ckb-daemon/profile_mouse.c File Reference

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

Include dependency graph for profile_mouse.c:



Functions

- int cmd_hwload_mouse (usbdevice ∗kb, usbmode ∗dummy1, int dummy2, int apply, const char ∗dummy3)
- int cmd_hwsave_mouse (usbdevice ∗kb, usbmode ∗dummy1, int dummy2, int dummy3, const char ∗dummy4)

9.39.1 Function Documentation

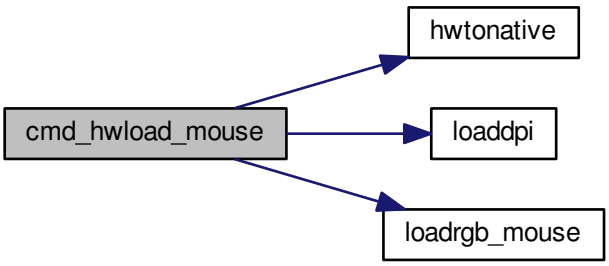
9.39.1.1 int cmd_hwload_mouse (usbdevice * *kb,* usbmode * *dummy1,* int *dummy2,* int *apply,* const char * *dummy3*)

Definition at line 6 of file profile_mouse.c.

References DELAY_LONG, hwprofile::dpi, usbdevice::hw, hwtonative(), hwprofile::id, hwprofile::light, loaddpi(), loadrgb_mouse(), MSG_SIZE, hwprofile::name, PR_NAME_LEN, usbdevice::profile, and usbrecv.

```
6                                                                                                                   {
7       (void)dummy1;
8       (void)dummy2;
9       (void)dummy3;
10
11      DELAY_LONG(kb);
12      hwprofile* hw = calloc(1, sizeof(hwprofile));
13      // Ask for profile and mode IDs
14      uchar data_pkt[2][MSG_SIZE] = {
15          { 0x0e, 0x15, 0x01, 0 },
16          { 0x0e, 0x16, 0x01, 0 }
17      };
18      uchar in_pkt[MSG_SIZE];
19      for(int i = 0; i <= 1; i++){
20          data_pkt[0][3] = i;
21          if(!usbrecv(kb, data_pkt[0], in_pkt)){
22              free(hw);
23              return -1;
24          }
25          memcpy(hw->id + i, in_pkt + 4, sizeof(usbid));
26      }
27      // Ask for profile and mode names
28      for(int i = 0; i <= 1; i++){
29          data_pkt[1][3] = i;
30          if(!usbrecv(kb, data_pkt[1],in_pkt)){
31              free(hw);
32              return -1;
33          }
34          memcpy(hw->name[i], in_pkt + 4, PR_NAME_LEN * 2);
35      }
36
37      // Load the RGB and DPI settings
38      if(loadrgb_mouse(kb, hw->light, 0)
39              || loaddpi(kb, hw->dpi, hw->light)){
40          free(hw);
41          return -1;
42      }
43
44      // Make the profile active (if requested)
45      if(apply)
46          hwtonative(kb->profile, hw, 1);
47      // Free the existing profile (if any)
48      free(kb->hw);
49      kb->hw = hw;
50      DELAY_LONG(kb);
51      return 0;
52  }
```

Here is the call graph for this function:



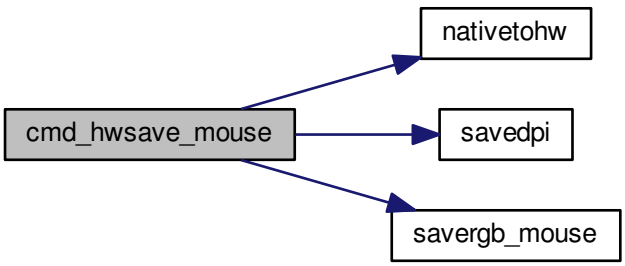
9.39.1.2 int cmd_hwsave_mouse (usbdevice * *kb*, usbmode * *dummy1*, int *dummy2*, int *dummy3*, const char * *dummy4*)

Definition at line 54 of file profile_mouse.c.

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

```
54                                                                                                  {
55     (void)dummy1;
56     (void)dummy2;
57     (void)dummy3;
58     (void)dummy4;
59
60     DELAY_LONG(kb);
61     hwprofile* hw = kb->hw;
62     if(!hw)
63         hw = kb->hw = calloc(1, sizeof(hwprofile));
64     nativetohw(kb->profile, hw, 1);
65     // Save the profile and mode names
66     uchar data_pkt[2][MSG_SIZE] = {
67         { 0x07, 0x16, 0x01, 0 },
68         { 0x07, 0x15, 0x01, 0 },
69     };
70     for(int i = 0; i <= 1; i++){
71         data_pkt[0][3] = i;
72         memcpy(data_pkt[0] + 4, hw->name[i], MD_NAME_LEN * 2);
73         if(!usbsend(kb, data_pkt[0], 1))
74             return -1;
75     }
76     // Save the IDs
77     for(int i = 0; i <= 1; i++){
78         data_pkt[1][3] = i;
79         memcpy(data_pkt[1] + 4, hw->id + i, sizeof(usbid));
80         if(!usbsend(kb, data_pkt[1], 1))
81             return -1;
82     }
83     // Save the RGB data for the non-DPI zones
84     if(savergb_mouse(kb, hw->light, 0))
85         return -1;
86     // Save the DPI data (also saves RGB for those states)
87     if(savedpi(kb, hw->dpi, hw->light))
88         return -1;
89     DELAY_LONG(kb);
90     return 0;
91 }
```

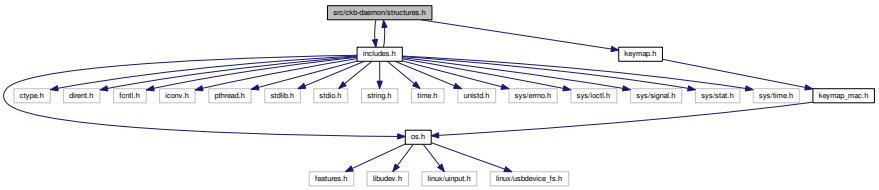
Here is the call graph for this function:



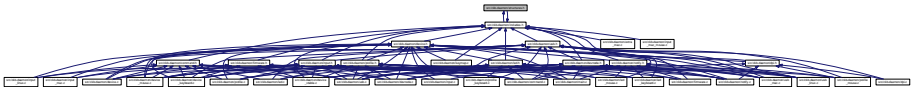
9.40 src/ckb-daemon/structures.h File Reference

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

Include dependency graph for structures.h:



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



Data Structures

- struct usbid
- struct macroaction
- struct keymacro
- struct binding
- struct dpiset
- struct lighting
- struct usbmode
- struct usbprofile
- struct hwprofile
- struct usbinput
- struct usbdevice

Macros

- #define SET_KEYBIT(array, index) do { (array)[(index) / 8] |= 1 << ((index) % 8); } while(0)
- #define CLEAR_KEYBIT(array, index) do { (array)[(index) / 8] &= ~(1 << ((index) % 8)); } while(0)
- #define I_NUM 1
- #define I_CAPS 2
- #define I_SCROLL 4
- #define OUTFIFO_MAX 10
- #define MACRO_MAX 1024
- #define DPI_COUNT 6
- #define LIFT_MIN 1
- #define LIFT_MAX 5
- #define MD_NAME_LEN 16
- #define PR_NAME_LEN 16
- #define MODE_COUNT 6
- #define HWMODE_K70 1
- #define HWMODE_K95 3
- #define HWMODE_MAX 3
- #define FEAT_RGB 0x001
- #define FEAT_MONOCHROME 0x002
- #define FEAT_POLLRATE 0x004

- #define FEAT_ADJRATE 0x008
- #define FEAT_BIND 0x010
- #define FEAT_NOTIFY 0x020
- #define FEAT_FWVERSION 0x040
- #define FEAT_FWUPDATE 0x080
- #define FEAT_HWLOAD 0x100
- #define FEAT_ANSI 0x200
- #define FEAT_ISO 0x400
- #define FEAT_MOUSEACCEL 0x800
- #define FEAT_COMMON (FEAT_BIND | FEAT_NOTIFY | FEAT_FWVERSION | FEAT_MOUSEACCEL | FEAT_HWLOAD)
- #define FEAT_STD_RGB (FEAT_COMMON | FEAT_RGB | FEAT_POLLRATE | FEAT_FWUPDATE)
- #define FEAT_STD_NRGB (FEAT_COMMON)
- #define FEAT_LMASK (FEAT_ANSI | FEAT_ISO)
- #define HAS_FEATURES(kb, feat) (((kb)->features & (feat)) == (feat))
- #define HAS_ANY_FEATURE(kb, feat) (!!((kb)->features & (feat)))
- #define NEEDS_FW_UPDATE(kb) ((kb)->fwversion == 0 && HAS_FEATURES((kb), FEAT_FWUPDATE | FEAT_FWVERSION))
- #define SCROLL_ACCELERATED 0
- #define SCROLL_MIN 1
- #define SCROLL_MAX 10
- #define KB_NAME_LEN 40
- #define SERIAL_LEN 34
- #define MSG_SIZE 64
- #define IFACE_MAX 4

Variables

- const union devcmd vtable_keyboard

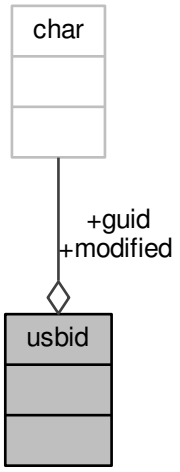
  RGB keyboard vtable holds functions for each device type.
- const union devcmd vtable_keyboard_nonrgb
- const union devcmd vtable_mouse

9.40.1 Data Structure Documentation

9.40.1.1 struct usbid

Definition at line 8 of file structures.h.

Collaboration diagram for usbid:



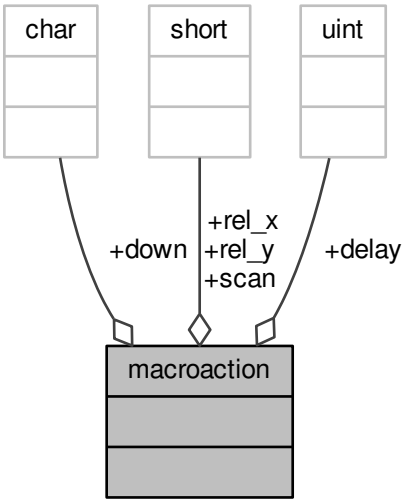
Data Fields

char	guid[16]	
char	modified[4]	

9.40.1.2 struct macroaction

Definition at line 27 of file structures.h.

Collaboration diagram for macroaction:



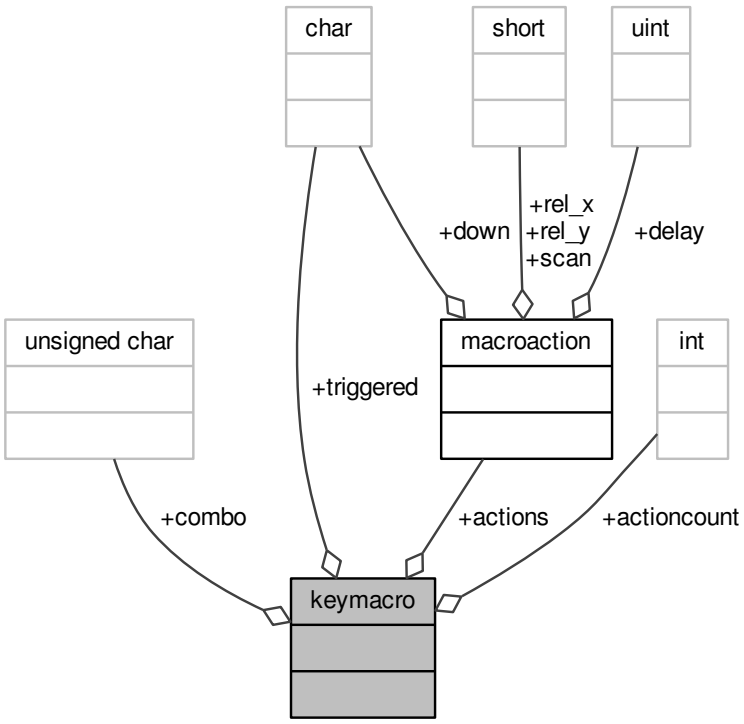
Data Fields

uint	delay	
char	down	
short	rel_x	
short	rel_y	
short	scan	

9.40.1.3 struct keymacro

Definition at line 35 of file structures.h.

Collaboration diagram for keymacro:



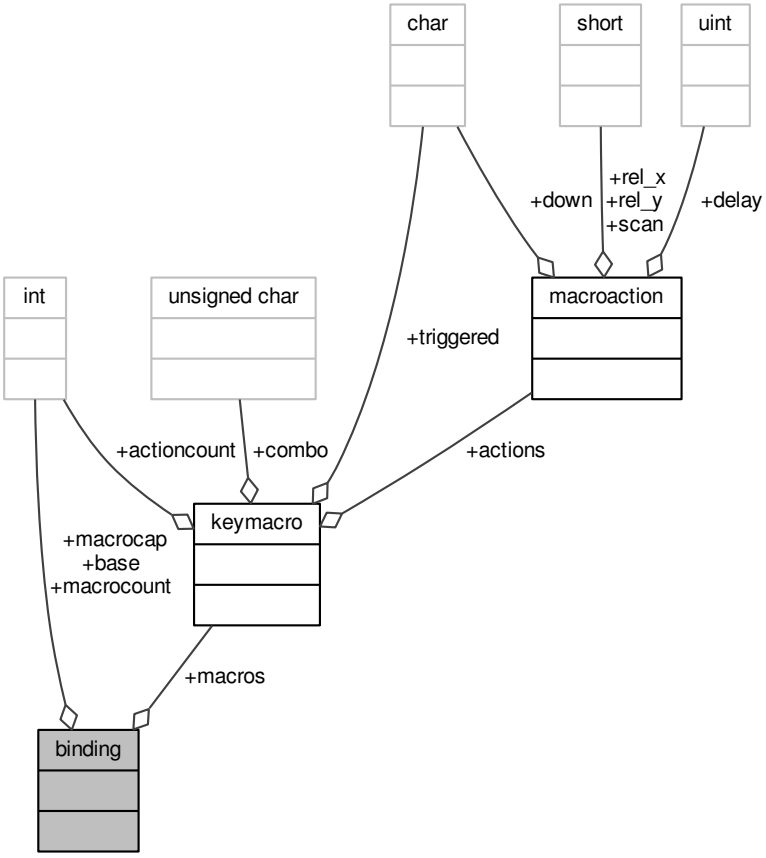
Data Fields

int	actioncount	
macroaction *	actions	
uchar	combo[(((((152+22+12)+25)+7)/8)]	
char	triggered	

9.40.1.4 struct binding

Definition at line 43 of file structures.h.

Collaboration diagram for binding:



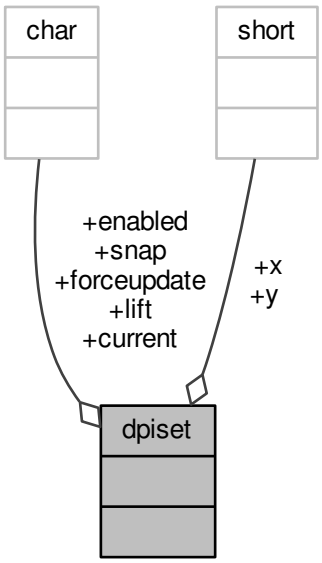
Data Fields

int	base[((152+22+12)+25)]	
int	macrocap	
int	macrocount	
keymacro *	macros	

9.40.1.5 struct dpiset

Definition at line 57 of file structures.h.

Collaboration diagram for dpiset:



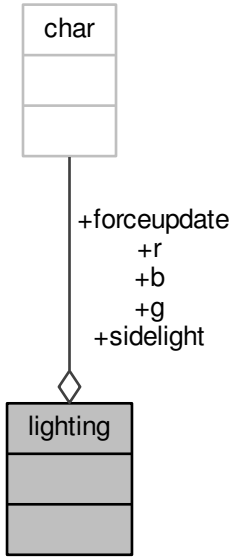
Data Fields

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

9.40.1.6 struct lighting

Definition at line 73 of file structures.h.

Collaboration diagram for lighting:



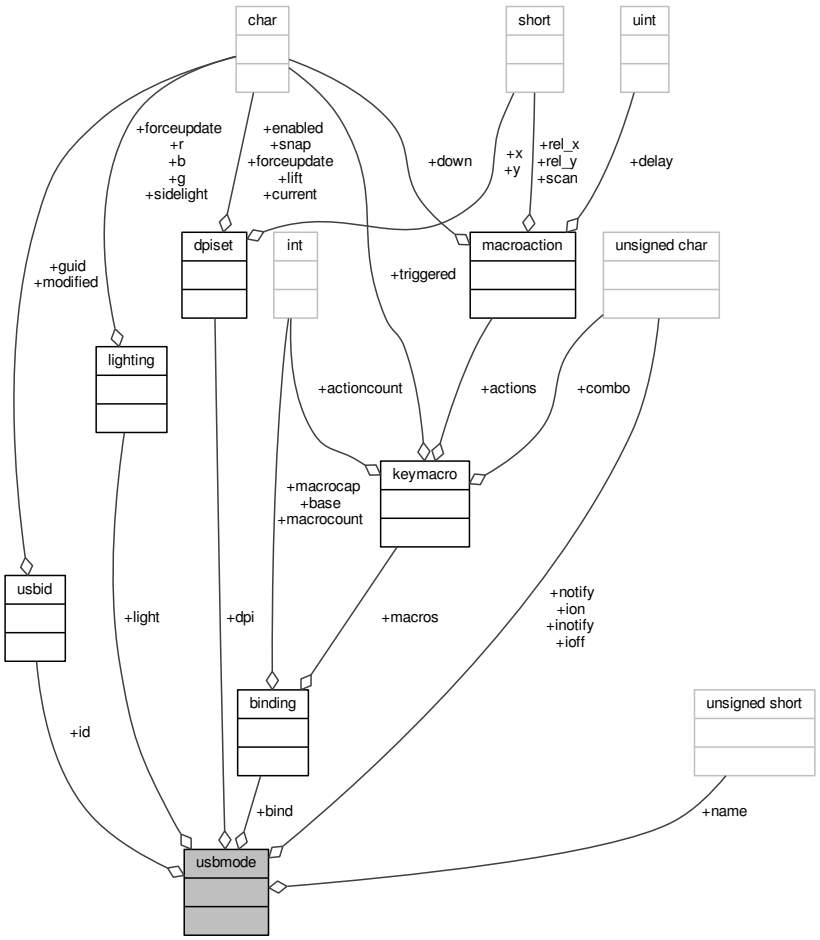
Data Fields

uchar	b[152+12]	
uchar	forceupdate	
uchar	g[152+12]	
uchar	r[152+12]	
uchar	sidelight	

9.40.1.7 struct usbmode

Definition at line 83 of file structures.h.

Collaboration diagram for usbmode:



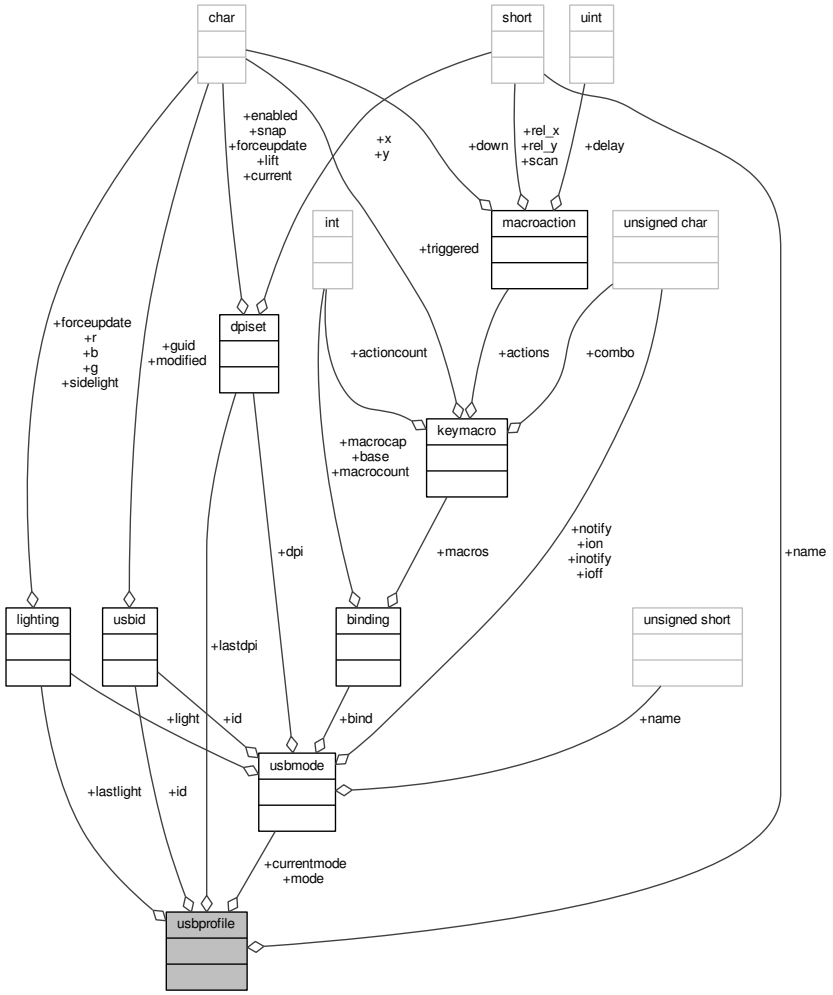
Data Fields

binding	bind	
dpiset	dpi	
usbid	id	
uchar	inotify[10]	
uchar	ioff	
uchar	ion	
lighting	light	
ushort	name[16]	
uchar	notify[10][((((152+22+12)+25)+7)/8)]	

9.40.1.8 struct usbprofile

Definition at line 101 of file structures.h.

Collaboration diagram for usbprofile:



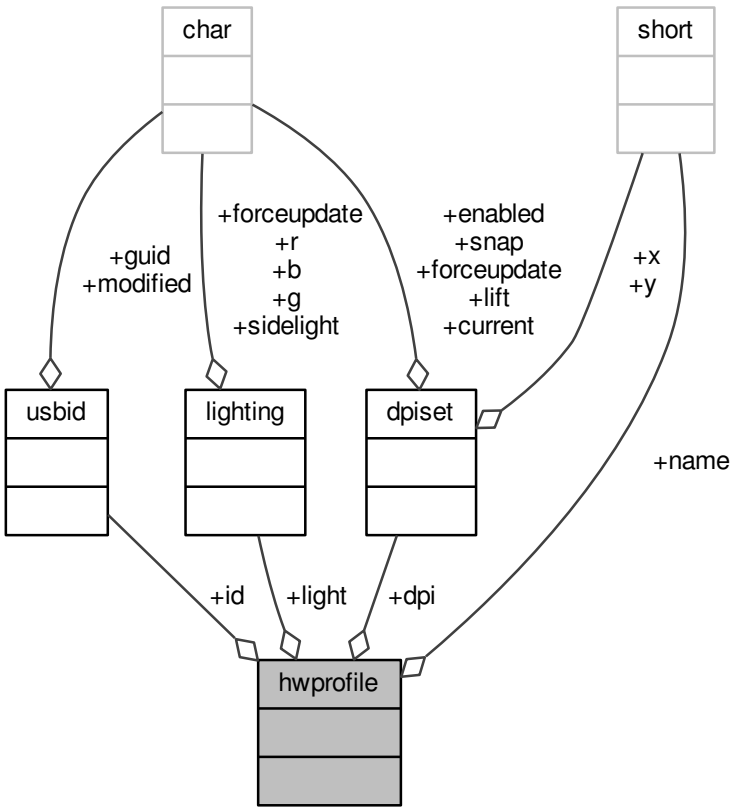
Data Fields

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

9.40.1.9 struct hwprofile

Definition at line 118 of file structures.h.

Collaboration diagram for hwprofile:



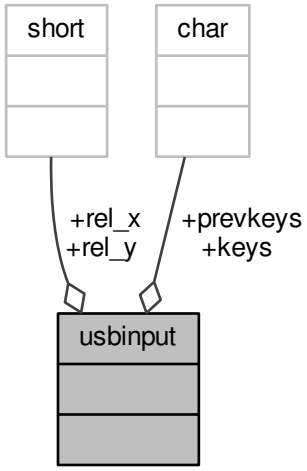
Data Fields

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

9.40.1.10 struct usbinput

Definition at line 129 of file structures.h.

Collaboration diagram for usbinput:



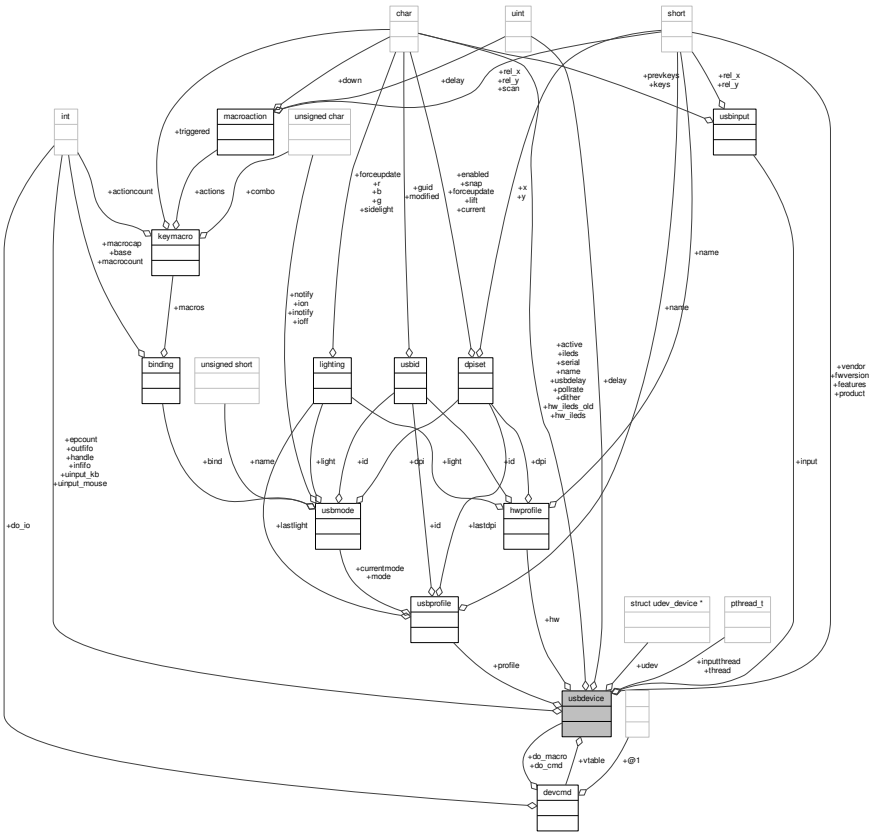
Data Fields

uchar	keys[((((152+22+12)+25)+7)/8)]	
uchar	prevkeys[((((152+22+12)+25)+7)/8)]	
short	rel_x	
short	rel_y	

9.40.1.11 struct usbdevice

Definition at line 178 of file structures.h.

Collaboration diagram for usbdevice:



Data Fields

char	active	
uint	delay	
char	dither	
int	epcount	
ushort	features	
ushort	fwversion	
int	handle	
hwprofile *	hw	
uchar	hw_ileds	
uchar	hw_ileds_old	
uchar	ileds	
int	infifo	
usbinput	input	
pthread_t	inputthread	
char	name[40+1]	
int	outfifo[10]	
char	pollrate	

short	product	
usbprofile *	profile	
char	serial[34]	
pthread_t	thread	
struct udev_device *	udev	
int	uinput_kb	
int	uinput_mouse	
char	usbdelay	
short	vendor	
const union devcmd *	vtable	

9.40.2 Macro Definition Documentation

9.40.2.1 #define CLEAR_KEYBIT(*array, index*) do { (array)[(index) / 8] &= ~(1 << ((index) % 8)); } while(0)

Definition at line 16 of file structures.h.

Referenced by cmd_notify(), corsair_mousecopy(), hid_kb_translate(), and hid_mouse_translate().

9.40.2.2 #define DPI_COUNT 6

Definition at line 54 of file structures.h.

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

9.40.2.3 #define FEAT_ADJRATE 0x008

Definition at line 139 of file structures.h.

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

9.40.2.4 #define FEAT_ANSI 0x200

Definition at line 146 of file structures.h.

Referenced by readcmd().

9.40.2.5 #define FEAT_BIND 0x010

Definition at line 140 of file structures.h.

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

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

Definition at line 151 of file structures.h.

9.40.2.7 #define FEAT_FWUPDATE 0x080

Definition at line 143 of file structures.h.

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

9.40.2.8 #define FEAT_FWVERSION 0x040

Definition at line 142 of file structures.h.

Referenced by _mkdevpath(), and _start_dev().

9.40.2.9 #define FEAT_HWLOAD 0x100

Definition at line 144 of file structures.h.

Referenced by _start_dev().

9.40.2.10 #define FEAT_ISO 0x400

Definition at line 147 of file structures.h.

Referenced by readcmd().

9.40.2.11 #define FEAT_LMASK (FEAT_ANSI | FEAT_ISO)

Definition at line 154 of file structures.h.

Referenced by readcmd().

9.40.2.12 #define FEAT_MONOCHROME 0x002

Definition at line 137 of file structures.h.

Referenced by _mkdevpath(), and _setupusb().

9.40.2.13 #define FEAT_MOUSEACCEL 0x800

Definition at line 148 of file structures.h.

Referenced by main(), and readcmd().

9.40.2.14 #define FEAT_NOTIFY 0x020

Definition at line 141 of file structures.h.

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

9.40.2.15 #define FEAT_POLLRATE 0x004

Definition at line 138 of file structures.h.

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

9.40.2.16 #define FEAT_RGB 0x001

Definition at line 136 of file structures.h.

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

9.40.2.17 #define FEAT_STD_NRGB (FEAT_COMMON)

Definition at line 153 of file structures.h.

Referenced by _setupusb().

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

Definition at line 152 of file structures.h.

Referenced by _setupusb().

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

Definition at line 158 of file structures.h.

9.40.2.20 #define HAS_FEATURES(*kb, feat*) (((kb)->features & (feat)) == (feat))

Definition at line 157 of file structures.h.

Referenced by _mkdevpath(), _start_dev(), cmd_fwupdate(), readcmd(), revertusb(), and usbunclaim().

9.40.2.21 #define HWMODE_K70 1

Definition at line 115 of file structures.h.

Referenced by cmd_hwload_kb(), and cmd_hwsave_kb().

9.40.2.22 #define HWMODE_K95 3

Definition at line 116 of file structures.h.

Referenced by cmd_hwload_kb(), and cmd_hwsave_kb().

9.40.2.23 #define HWMODE_MAX 3

Definition at line 117 of file structures.h.

9.40.2.24 #define I_CAPS 2

Definition at line 20 of file structures.h.

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

9.40.2.25 #define I_NUM 1

Definition at line 19 of file structures.h.

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

9.40.2.26 #define I_SCROLL 4

Definition at line 21 of file structures.h.

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

9.40.2.27 #define IFACE_MAX 4

Definition at line 177 of file structures.h.

9.40.2.28 #define KB_NAME_LEN 40

Definition at line 174 of file structures.h.

Referenced by _setupusb(), and os_setupusb().

9.40.2.29 #define LIFT_MAX 5

Definition at line 56 of file structures.h.

Referenced by cmd_lift(), and loaddpi().

9.40.2.30 #define LIFT_MIN 1

Definition at line 55 of file structures.h.

Referenced by cmd_lift(), and loaddpi().

9.40.2.31 #define MACRO_MAX 1024

Definition at line 51 of file structures.h.

Referenced by _cmd_macro().

9.40.2.32 #define MD_NAME_LEN 16

Definition at line 82 of file structures.h.

Referenced by cmd_hwsave_kb(), cmd_hwsave_mouse(), cmd_name(), gethwmodename(), gethwprofilename(), getmodename(), hwloadmode(), hwtonative(), and nativetohw().

9.40.2.33 #define MODE_COUNT 6

Definition at line 100 of file structures.h.

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

9.40.2.34 #define MSG_SIZE 64

Definition at line 176 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().

9.40.2.35 #define NEEDS_FW_UPDATE(*kb*) ((kb)->fwversion == 0 && HAS_FEATURES((kb), FEAT_FWUPDATE | FEAT_FWVERSION))

Definition at line 161 of file structures.h.

Referenced by _start_dev(), readcmd(), revertusb(), setactive_kb(), and setactive_mouse().

9.40.2.36 #define OUTFIFO_MAX 10

Definition at line 24 of file structures.h.

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

9.40.2.37 #define PR_NAME_LEN 16

Definition at line 99 of file structures.h.

Referenced by cmd_hwload_kb(), cmd_hwload_mouse(), cmd_profilename(), getprofilename(), hwtonative(), and nativetohw().

9.40.2.38 #define SCROLL_ACCELERATED 0

Definition at line 164 of file structures.h.

Referenced by readcmd().

9.40.2.39 #define SCROLL_MAX 10

Definition at line 166 of file structures.h.

Referenced by readcmd().

9.40.2.40 #define SCROLL_MIN 1

Definition at line 165 of file structures.h.

Referenced by readcmd().

9.40.2.41 #define SERIAL_LEN 34

Definition at line 175 of file structures.h.

Referenced by _setupusb(), and os_setupusb().

9.40.2.42 #define SET_KEYBIT(*array, index*) do { (array)[(index) / 8] |= 1 << ((index) % 8); } while(0)

Definition at line 15 of file structures.h.

Referenced by _cmd_macro(), cmd_notify(), corsair_mousecopy(), hid_kb_translate(), and hid_mouse_translate().

9.40.3 Variable Documentation

9.40.3.1 const union devcmd vtable_keyboard

Definition at line 52 of file device_vtable.c.

Referenced by get_vtable().

9.40.3.2 const union devcmd vtable_keyboard_nonrgb

Definition at line 99 of file device_vtable.c.

Referenced by get_vtable().

9.40.3.3 const union devcmd vtable_mouse

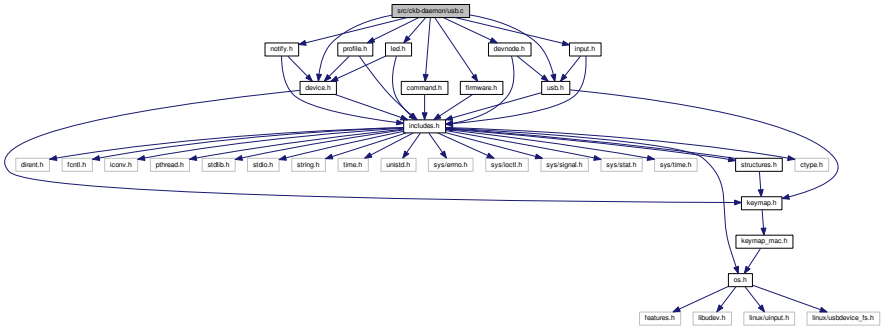
Definition at line 146 of file device_vtable.c.

Referenced by get_vtable().

9.41 src/ckb-daemon/usb.c File Reference

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

Include dependency graph for usb.c:



Functions

- const char * vendor_str (short vendor)

  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

  hwload_mode is defined in device.c

9.41.1 Function Documentation

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

_resetusb Reset a USB device.

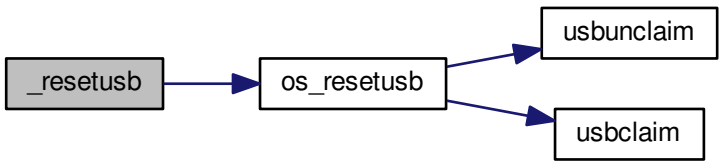
First reset the device via os_resetusb() after a long delay (it may send something to the host). If this worked (retval == 0), give the device another long delay Then perform the initialization via the device specific start() function entry in kb->vtable and if this is successful also, return the result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

Definition at line 436 of file usb.c.

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

```
436                                                     {
437     // Perform a USB reset
438     DELAY_LONG(kb);
439     int res = os_resetusb(kb, file, line);
440     if(res)
441         return res;
442     DELAY_LONG(kb);
443     // Re-initialize the device
444     if(kb->vtable->start(kb, kb->active) != 0)
445         return -1;
446     if(kb->vtable->updatergb(kb, 1) != 0)
447         return -1;
448     return 0;
449 }
```

Here is the call graph for this function:



9.41.1.2 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 ptread_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 224 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().

```
224                                     {
237     usbdevice* kb = context;
238     // Set standard fields
239     short vendor = kb->vendor, product = kb->product;
240     const devcmd* vt = kb->vtable = get_vtable(vendor, product);
241     kb->features = (IS_RGB(vendor, product) ? FEAT_STD_RGB :
    FEAT_STD_NRGB) & features_mask;
242     if(IS_MOUSE(vendor, product)) kb->features |= FEAT_ADJRATE;
243     if(IS_MONOCHROME(vendor, product)) kb->features |=
    FEAT_MONOCHROME;
244     kb->usbdelay = USB_DELAY_DEFAULT;
245
246     // Perform OS-specific setup
250     DELAY_LONG(kb);
251
257     if(os_setupusb(kb))
258         goto fail;
259
265     // Make up a device name and serial if they weren't assigned
266     if(!kb->serial[0])
267         snprintf(kb->serial, SERIAL_LEN, "%04x:%04x-NoID", kb->
    vendor, kb->product);
268     if(!kb->name[0])
269         snprintf(kb->name, KB_NAME_LEN, "%s %s", vendor_str(kb->
    vendor), product_str(kb->product));
270
271     // Set up an input device for key events
279     if(os_inputopen(kb))
280         goto fail;
284     if(pthread_create(&kb->inputthread, 0, os_inputmain, kb))
285         goto fail;
286     pthread_detach(kb->inputthread);
292     if(os_setupindicators(kb))
293         goto fail;
294
295     // Set up device
308     vt->allocprofile(kb);
```

```
319     vt->updateindicators(kb, 1);
324     pthread_mutex_unlock(imutex(kb));
358     if(vt->start(kb, 0) && usb_tryreset(kb))
359         goto fail_noinput;
365     // Make /dev path
366     if(mkdevpath(kb))
367         goto fail_noinput;
373     // Finished. Enter main loop
374     int index = INDEX_OF(kb, keyboard);
375     ckb_info("Setup finished for %s%d\n", devpath, index);
376     updateconnected();
379     return devmain(kb);
382     fail:
383     pthread_mutex_unlock(imutex(kb));
384     fail_noinput:
385     closeusb(kb);
386     pthread_mutex_unlock(dmutex(kb));
387     return 0;
388 }
```

Here is the call graph for this function:

Here is the caller graph for this function:

9.41.1.3 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. < Synchonization between macro and color information

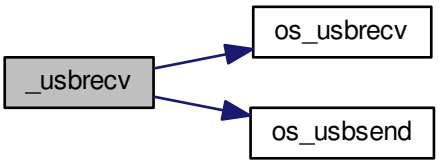
Definition at line 611 of file usb.c.

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

```
611                                                                                                 {
612     // Try a maximum of 5 times
613     for (int try = 0; try < 5; try++) {
614         // Send the output message
615         pthread_mutex_lock(mmutex(kb));
616         DELAY_SHORT(kb);
617         int res = os_usbsend(kb, out_msg, 1, file, line);
618         pthread_mutex_unlock(mmutex(kb));
619         if (res == 0)
620             return 0;
621         else if (res == -1) {
622             // Retry on temporary failure
623             if (reset_stop)
624                 return 0;
625             DELAY_LONG(kb);
626             continue;
627         }
628         // Wait for the response
629         DELAY_MEDIUM(kb);
630         res = os_usbrecv(kb, in_msg, file, line);
631         if(res == 0)
632             return 0;
633         else if(res != -1)
634             return res;
635         if(reset_stop || hwload_mode != 2)
636             return 0;
637         DELAY_LONG(kb);
638     }
639     // Give up
640     ckb_err_fn("Too many send/recv failures. Dropping.\n", file, line);
641     return 0;
```

```
642 }
```

Here is the call graph for this function:



9.41.1.4 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 (...). < Synchonization between macro and color information

Definition at line 542 of file usb.c.

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

```
542                                                                                                    {
543     int total_sent = 0;
544     for(int i = 0; i < count; i++){
545         // Send each message via the OS function
546         while(1){
547             pthread_mutex_lock(mmutex(kb));
548             DELAY_SHORT(kb);
549             int res = os_usbsend(kb, messages + i * MSG_SIZE, 0, file, line);
550             pthread_mutex_unlock(mmutex(kb));
551             if(res == 0)
552                 return 0;
553             else if(res != -1){
554                 total_sent += res;
555                 break;
556             }
557             // Stop immediately if the program is shutting down or hardware load is set to tryonce
558             if(reset_stop || hwload_mode != 2)
559                 return 0;
560             // Retry as long as the result is temporary failure
561             DELAY_LONG(kb);
562         }
563     }
564     return total_sent;
565 }
```

Here is the call graph for this function:

9.41.1.5 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).

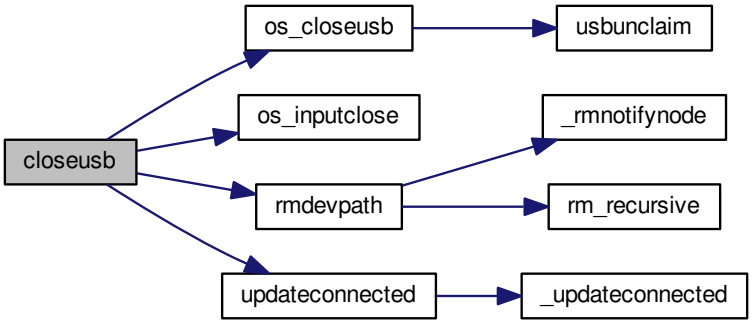
Definition at line 687 of file usb.c.

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

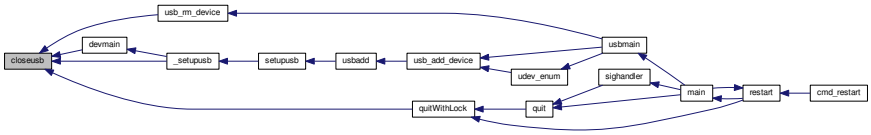
Referenced by _setupusb(), devmain(), quitWithLock(), and usb_rm_device().

```
687                             {
688     pthread_mutex_lock(imutex(kb));
689     if(kb->handle){
690         int index = INDEX_OF(kb, keyboard);
691         ckb_info("Disconnecting %s%d\n", devpath, index);
692         os_inputclose(kb);
693         updateconnected();
694         // Close USB device
695         os_closeusb(kb);
696     } else
697         updateconnected();
698     rmdevpath(kb);
699 
700     // Wait for thread to close
701     pthread_mutex_unlock(imutex(kb));
702     pthread_mutex_unlock(dmutex(kb));
703     pthread_join(kb->thread, 0);
704     pthread_mutex_lock(dmutex(kb));
705 
706     // Delete the profile and the control path
707     if(!kb->vtable)
708         return 0;
709     kb->vtable->freeprofile(kb);
710     memset(kb, 0, sizeof(usbdevice));
711     return 0;
712 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



9.41.1.6 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 145 of file usb.c.

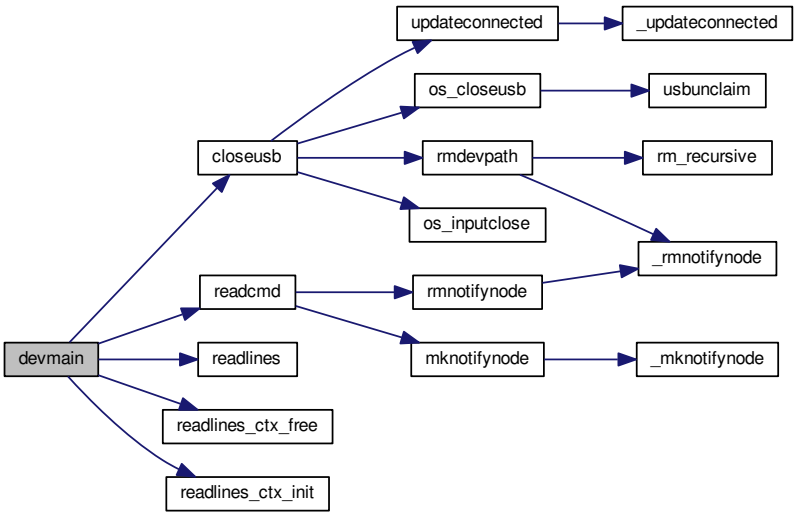
References closeusb(), dmutex, usbdevice::infifo, IS_CONNECTED, readcmd(), readlines(), readlines_ctx_free(), and readlines_ctx_init().

Referenced by _setupusb().

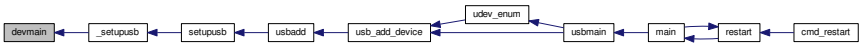
```
145                                     {
147     int kbfifo = kb->infifo - 1;
150     readlines_ctx linectx;
151     readlines_ctx_init(&linectx);
156     while(1){
163         pthread_mutex_unlock(dmutex(kb));
164         // Read from FIFO
165         const char* line;
166         int lines = readlines(kbfifo, linectx, &line);
167         pthread_mutex_lock(dmutex(kb));
168         // End thread when the handle is removed
169         if(!IS_CONNECTED(kb))
170             break;
174         if(lines){
177             if(readcmd(kb, line)){
183                 // USB transfer failed; destroy device
184                 closeusb(kb);
185                 break;
```

```
186             }
187         }
188     }
189     pthread_mutex_unlock(dmutex(kb));
192     readlines_ctx_free(linectx);
193     return 0;
194 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



9.41.1.7 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?

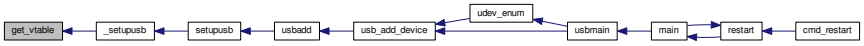
Definition at line 112 of file usb.c.

References IS_MOUSE, IS_RGB, vtable_keyboard, vtable_keyboard_nonrgb, and vtable_mouse.

Referenced by _setupusb().

```
112                                                                    {
113     return IS_MOUSE(vendor, product) ? &vtable_mouse :
      IS_RGB(vendor, product) ? &vtable_keyboard : &
      vtable_keyboard_nonrgb;
114 }
```

Here is the caller graph for this function:



9.41.1.8 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.

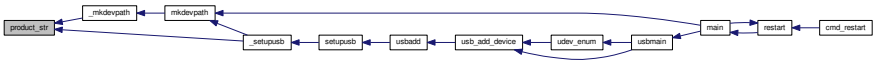
References P_GLAIVE, P_HARPOON, P_K63_NRGB, P_K65, P_K65_LUX, P_K65_NRGB, P_K65_RFIRE, P-_K68, P_K70, P_K70_LUX, P_K70_LUX_NRGB, P_K70_NRGB, P_K70_RFIRE, P_K70_RFIRE_NRGB, P_K95, P_K95_NRGB, P_K95_PLATINUM, P_M65, P_M65_PRO, P_SABRE_L, P_SABRE_N, P_SABRE_O, P_SABRE-_O2, P_SCIMITAR, P_SCIMITAR_PRO, P_STRAFE, P_STRAFE_NRGB, and P_STRAFE_NRGB_2.

Referenced by _mkdevpath(), and _setupusb().

```
70                                          {
71     if(product == P_K95 || product == P_K95_NRGB)
72         return "k95";
73     if(product == P_K95_PLATINUM)
74         return "k95p";
75     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)
76         return "k70";
77     if(product == P_K68)
78         return "k68";
79     if(product == P_K65 || product == P_K65_NRGB || product ==
```

```
    P_K65_LUX || product == P_K65_RFIRE)
80          return "k65";
81      if(product == P_K63_NRGB)
82          return "k63";
83      if(product == P_STRAFE || product == P_STRAFE_NRGB || product ==
    P_STRAFE_NRGB_2)
84          return "strafe";
85      if(product == P_M65 || product == P_M65_PRO)
86          return "m65";
87      if(product == P_SABRE_O || product == P_SABRE_L || product ==
    P_SABRE_N || product == P_SABRE_O2)
88          return "sabre";
89      if(product == P_SCIMITAR || product == P_SCIMITAR_PRO)
90          return "scimitar";
91      if(product == P_HARPOON)
92          return "harpoon";
93      if(product == P_GLAIVE)
94          return "glaive";
95      return "";
96  }
```

Here is the caller graph for this function:



9.41.1.9 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 417 of file usb.c.

References FEAT_RGB, HAS_FEATURES, NEEDS_FW_UPDATE, NK95_HWON, nk95cmd, and setactive.

Referenced by quitWithLock().

```
417                             {
418      if(NEEDS_FW_UPDATE(kb))
419          return 0;
420      if(!HAS_FEATURES(kb, FEAT_RGB)){
421          nk95cmd(kb, NK95_HWON);
422          return 0;
423      }
424      if(setactive(kb, 0))
425          return -1;
426      return 0;
427  }
```

Here is the caller graph for this function:

9.41.1.10 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.

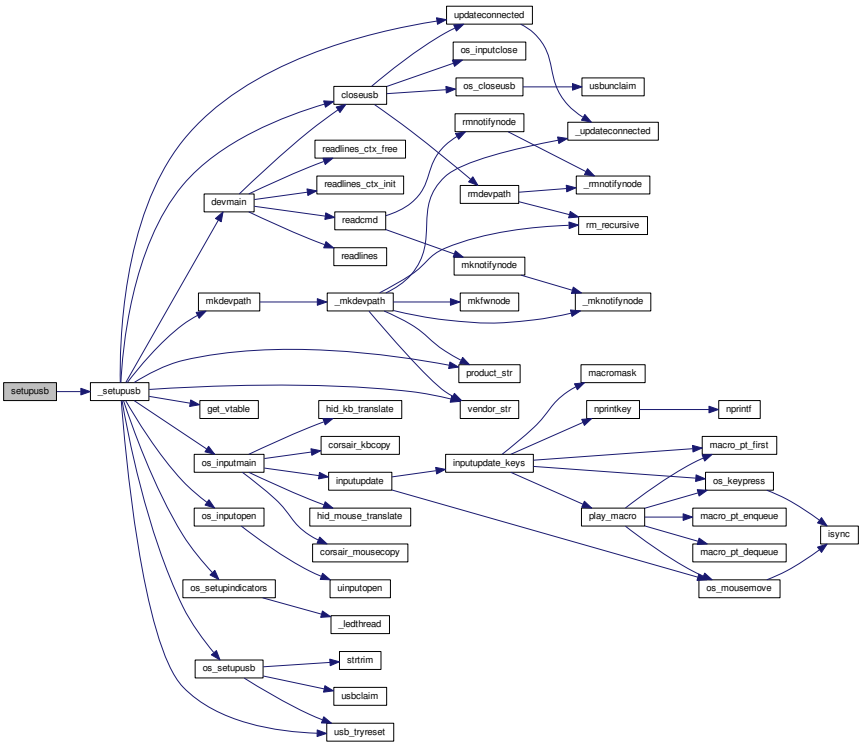
Definition at line 396 of file usb.c.

References _setupusb(), ckb_err, imutex, and usbdevice::thread.

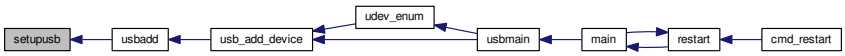
Referenced by usbadd().

```
396                         {
397     pthread_mutex_lock(imutex(kb));
398     if(pthread_create(&kb->thread, 0, _setupusb, kb))
399         ckb_err("Failed to create USB thread\n");
400 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



9.41.1.11 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?

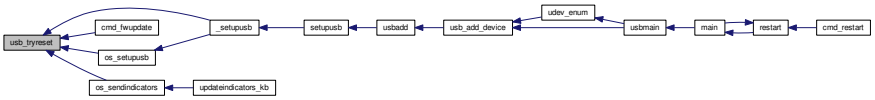
Definition at line 475 of file usb.c.

References ckb_err, ckb_info, reset_stop, and resetusb.

Referenced by _setupusb(), cmd_fwupdate(), os_sendindicators(), and os_setupusb().

```
475                                {
476     if(reset_stop)
477         return -1;
478     ckb_info("Attempting reset...\n");
479     while(1){
480         int res = resetusb(kb);
481         if(!res){
482             ckb_info("Reset success\n");
483             return 0;
484         }
485         if(res == -2 || reset_stop)
486             break;
487     }
488     ckb_err("Reset failed. Disconnecting.\n");
489     return -1;
490 }
```

Here is the caller graph for this function:



9.41.1.12 const char* vendor_str (short *vendor*)

uncomment to see USB packets sent to the device

vendor_str returns "corsair" if 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".

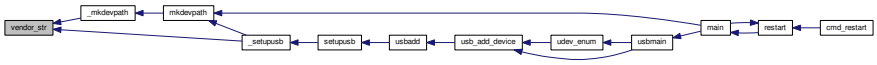
Definition at line 43 of file usb.c.

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:



9.41.2 Variable Documentation

9.41.2.1 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().

9.41.2.2 int hwload_mode

hwload_mode is defined in device.c

Definition at line 7 of file device.c.

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

9.41.2.3 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().

9.41.2.4 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.

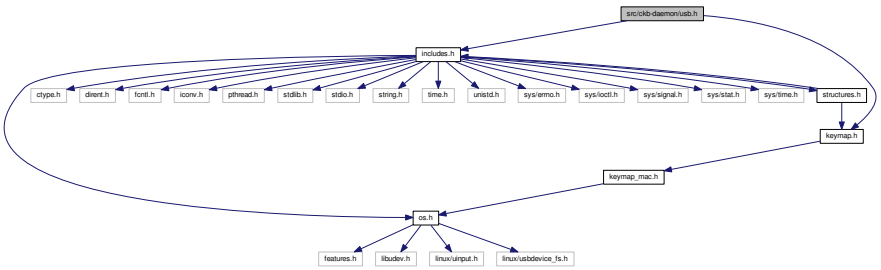
Definition at line 17 of file usb.c.

9.42 src/ckb-daemon/usb.h File Reference

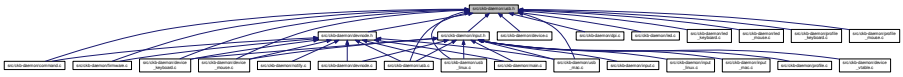
Definitions for using USB interface.

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

Include dependency graph for usb.h:



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



Macros

- #define V_CORSAIR 0x1b1c

  For the following Defines please see "Detailed Description".
- #define V_CORSAIR_STR "1b1c"
- #define P_K63_NRGB 0x1b40
- #define P_K63_NRGB_STR "1b40"
- #define IS_K63(kb) ((kb)->vendor == V_CORSAIR && (kb)->product == P_K63_NRGB)
- #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_K68 0x1b3f
- #define P_K68_STR "1b3f"
- #define IS_K68(kb) ((kb)->vendor == V_CORSAIR && (kb)->product == P_K68)
- #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 P_STRAFE_NRGB_2 0x1b44
- #define P_STRAFE_NRGB_2_STR "1b44"
- #define IS_STRAFE(kb) ((kb)->vendor == V_CORSAIR && ((kb)->product == P_STRAFE || (kb)->product == P_STRAFE_NRGB || (kb)->product == P_STRAFE_NRGB_2))
- #define P_M65 0x1b12
- #define P_M65_STR "1b12"
- #define P_M65_PRO 0x1b2e
- #define P_M65_PRO_STR "1b2e"
- #define IS_M65(kb) ((kb)->vendor == V_CORSAIR && ((kb)->product == P_M65 || (kb)->product == P_M65_PRO))
- #define P_SABRE_O 0x1b14 /* optical */
- #define P_SABRE_O_STR "1b14"
- #define P_SABRE_L 0x1b19 /* laser */
- #define P_SABRE_L_STR "1b19"
- #define P_SABRE_N 0x1b2f /* new? */
- #define P_SABRE_N_STR "1b2f"
- #define P_SABRE_O2 0x1b32 /* Observed on a CH-9000111-EU model SABRE */

- #define P_SABRE_O2_STR "1b32"
- #define IS_SABRE(kb) ((kb)->vendor == V_CORSAIR && ((kb)->product == P_SABRE_O || (kb)->product == P_SABRE_L || (kb)->product == P_SABRE_N || (kb)->product == P_SABRE_O2))
- #define P_SCIMITAR 0x1b1e
- #define P_SCIMITAR_STR "1b1e"
- #define P_SCIMITAR_PRO 0x1b3e
- #define P_SCIMITAR_PRO_STR "1b3e"
- #define IS_SCIMITAR(kb) ((kb)->vendor == V_CORSAIR && ((kb)->product == P_SCIMITAR || (kb)->product == P_SCIMITAR_PRO))
- #define P_HARPOON 0x1b3c
- #define P_HARPOON_STR "1b3c"
- #define IS_HARPOON(kb) ((kb)->vendor == V_CORSAIR && (kb)->product == P_HARPOON)
- #define P_GLAIVE 0x1b34
- #define P_GLAIVE_STR "1b34"
- #define IS_GLAIVE(kb) ((kb)->vendor == V_CORSAIR && (kb)->product == P_GLAIVE)
- #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_K68) || (product) == (P_STRAFE_NRGB) || (product) == (P_STRAFE_NRGB_2)))

  *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) || (product) == (P_GLAIVE) || (product) == (P_HARPOON)))

  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 IS_PLATINUM(kb) ((kb)->vendor == V_CORSAIR && ((kb)->product == P_K95_PLATINUM))

  Used to apply quirks and features to the PLATINUM devices.
- #define IS_V2_OVERRIDE(kb) (IS_PLATINUM(kb) || IS_K63(kb) || IS_K68(kb) || IS_HARPOON(kb) || IS_GLAIVE(kb) || (kb)->product == P_STRAFE_NRGB_2)

  Used when a device has a firmware with a low version number that uses the new protocol.
- #define DELAY_SHORT(kb) clock_nanosleep(CLOCK_MONOTONIC, 0, &(struct timespec) {.tv_nsec = ((int) (kb->usbdelay)) ∗ 1000000}, NULL)

  USB delays for when the keyboards get picky about timing That was the original comment, but it is used anytime.
- #define DELAY_MEDIUM(kb) clock_nanosleep(CLOCK_MONOTONIC, 0, &(struct timespec) {.tv_nsec = ((int) (kb->usbdelay)) ∗ 10000000}, NULL)

  the medium delay is used after sending a command before waiting for the answer.
- #define DELAY_LONG(kb) clock_nanosleep(CLOCK_MONOTONIC, 0, &(struct timespec) {.tv_nsec = 100000000}, NULL)

  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 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()

9.42.1 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 K63 Non RGB Keyboard. No other K63 is known so far. 2 | the K65-like keyboards, regardless of their properties (RGB, ...). | In summary, they can be queried using the macro IS_K65(). 3 | K68 keyboard | IS_K68(). 4 | the K70-like Keyboards with all their configuration types | summarized by IS_K70(). 5 | the K95 series keyboards | collected with the macro IS_K95(). 6 | strafe keyboards | IS_STRAFE() 7 | M65 mice with and without RGB | IS_M65() 8 | Sabre mice | IS_SABRE() 9 | Scimitar mice | IS_SCIMITAR() 10| Harpoon mice | IS_HARPOON() 11| Glaive mice | IS_GLAIVE()

Definition in file usb.h.

9.42.2 Macro Definition Documentation

9.42.2.1 #define DELAY_LONG(*kb*) clock_nanosleep(CLOCK_MONOTONIC, 0, &(struct timespec) {.tv_nsec = 100000000}, NULL)

Definition at line 186 of file usb.h.

Referenced by _resetusb(), _setupusb(), _usbrecv(), _usbsend(), cmd_hwload_kb(), cmd_hwload_mouse(), cmd_hwsave_kb(), and cmd_hwsave_mouse().

9.42.2.2 #define DELAY_MEDIUM(*kb*) clock_nanosleep(CLOCK_MONOTONIC, 0, &(struct timespec) {.tv_nsec = ((int)(kb->usbdelay)) * 10000000}, NULL)

Definition at line 182 of file usb.h.

Referenced by _usbrecv(), and setactive_kb().

9.42.2.3 #define DELAY_SHORT(kb) clock_nanosleep(CLOCK_MONOTONIC, 0, &(struct timespec) {.tv_nsec = ((int)(kb->usbdelay)) * 1000000}, NULL)

The short delay is used before any send or receive

Definition at line 178 of file usb.h.

Referenced by _usbrecv(), _usbsend(), and updateindicators_kb().

9.42.2.4 #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 160 of file usb.h.

Referenced by readcmd(), and updatergb_kb().

9.42.2.5 #define IS_GLAIVE(kb) ((kb)->vendor == V_CORSAIR && (kb)->product == P_GLAIVE)

Definition at line 121 of file usb.h.

Referenced by updatergb_mouse().

9.42.2.6 #define IS_HARPOON(kb) ((kb)->vendor == V_CORSAIR && (kb)->product == P_HARPOON)

Definition at line 117 of file usb.h.

9.42.2.7 #define IS_K63(kb) ((kb)->vendor == V_CORSAIR && (kb)->product == P_K63_NRGB)

Definition at line 47 of file usb.h.

Referenced by has_key().

9.42.2.8 #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 57 of file usb.h.

Referenced by has_key().

9.42.2.9 #define IS_K68(kb) ((kb)->vendor == V_CORSAIR && (kb)->product == P_K68)

Definition at line 61 of file usb.h.

9.42.2.10 #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 75 of file usb.h.

9.42.2.11 #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 83 of file usb.h.

Referenced by cmd_hwload_kb(), cmd_hwsave_kb(), and has_key().

9.42.2.12 #define IS_M65(*kb*) ((kb)->vendor == V_CORSAIR && ((kb)->product == P_M65 || (kb)->product == P_M65_PRO))

Definition at line 97 of file usb.h.

Referenced by isblack().

9.42.2.13 #define IS_MONOCHROME(*vendor, product*) ((vendor) == (V_CORSAIR) && ((product) == (P_K68) || (product) == (P_STRAFE_NRGB) || (product) == (P_STRAFE_NRGB_2)))

Definition at line 151 of file usb.h.

Referenced by _setupusb().

9.42.2.14 #define IS_MONOCHROME_DEV(*kb*) IS_MONOCHROME((kb)->vendor, (kb)->product)

Definition at line 157 of file usb.h.

9.42.2.15 #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) || (product) == (P_GLAIVE) || (product) == (P_HARPOON)))

Definition at line 163 of file usb.h.

Referenced by _setupusb(), get_vtable(), has_key(), and os_inputmain().

9.42.2.16 #define IS_MOUSE_DEV(*kb*) IS_MOUSE((kb)->vendor, (kb)->product)

Definition at line 166 of file usb.h.

Referenced by readcmd().

9.42.2.17 #define IS_PLATINUM(*kb*) ((kb)->vendor == V_CORSAIR && ((kb)->product == P_K95_PLATINUM))

Definition at line 169 of file usb.h.

9.42.2.18 #define IS_RGB(*vendor, product*) ((vendor) == (V_CORSAIR) && (product) != (P_K65_NRGB) && (product) != (P_K70_NRGB) && (product) != (P_K95_NRGB))

Definition at line 146 of file usb.h.

Referenced by _setupusb(), get_vtable(), and os_inputmain().

9.42.2.19 #define IS_RGB_DEV(*kb*) IS_RGB((kb)->vendor, (kb)->product)

Definition at line 154 of file usb.h.

9.42.2.20 #define IS_SABRE(*kb*) ((kb)->vendor == V_CORSAIR && ((kb)->product == P_SABRE_O || (kb)->product == P_SABRE_L || (kb)->product == P_SABRE_N || (kb)->product == P_SABRE_O2))

Definition at line 107 of file usb.h.

Referenced by has_key(), loadrgb_mouse(), and savergb_mouse().

9.42.2.21 #define IS_SCIMITAR(*kb*) ((kb)->vendor == V_CORSAIR && ((kb)->product == P_SCIMITAR || (kb)->product == P_SCIMITAR_PRO))

Definition at line 113 of file usb.h.

Referenced by has_key(), loadrgb_mouse(), and savergb_mouse().

9.42.2.22 #define IS_STRAFE(*kb*) ((kb)->vendor == V_CORSAIR && ((kb)->product == P_STRAFE || (kb)->product == P_STRAFE_NRGB || (kb)->product == P_STRAFE_NRGB_2))

Definition at line 91 of file usb.h.

Referenced by savergb_kb().

9.42.2.23 #define IS_V2_OVERRIDE(*kb*) (IS_PLATINUM(kb) || IS_K63(kb) || IS_K68(kb) || IS_HARPOON(kb) || IS_GLAIVE(kb) || (kb)->product == P_STRAFE_NRGB_2)

Definition at line 172 of file usb.h.

Referenced by loadrgb_kb(), os_usbsend(), and savergb_kb().

9.42.2.24 #define NK95_HWOFF 0x020030

See Also

usb2.0 documentation for details. Set Hardware playback off

Definition at line 333 of file usb.h.

Referenced by start_kb_nrgb().

9.42.2.25 #define NK95_HWON 0x020001

Definition at line 336 of file usb.h.

Referenced by revertusb().

9.42.2.26 #define NK95_M1 0x140001

Definition at line 339 of file usb.h.

Referenced by setmodeindex_nrgb().

9.42.2.27 #define NK95_M2 0x140002

Definition at line 342 of file usb.h.

Referenced by setmodeindex_nrgb().

9.42.2.28 #define NK95_M3 0x140003

Definition at line 345 of file usb.h.

Referenced by setmodeindex_nrgb().

9.42.2.29 #define nk95cmd(*kb, command*) _nk95cmd(kb, (command) >> 16 & 0xFF, (command) & 0xFFFF, __FILE_NOPATH__, __LINE__)

Definition at line 328 of file usb.h.

Referenced by revertusb(), setmodeindex_nrgb(), and start_kb_nrgb().

9.42.2.30 #define P_GLAIVE 0x1b34

Definition at line 119 of file usb.h.

Referenced by product_str().

9.42.2.31 #define P_GLAIVE_STR "1b34"

Definition at line 120 of file usb.h.

9.42.2.32 #define P_HARPOON 0x1b3c

Definition at line 115 of file usb.h.

Referenced by product_str().

9.42.2.33 #define P_HARPOON_STR "1b3c"

Definition at line 116 of file usb.h.

9.42.2.34 #define P_K63_NRGB 0x1b40

Definition at line 45 of file usb.h.

Referenced by product_str().

9.42.2.35 #define P_K63_NRGB_STR "1b40"

Definition at line 46 of file usb.h.

9.42.2.36 #define P_K65 0x1b17

Definition at line 49 of file usb.h.

Referenced by product_str().

9.42.2.37 #define P_K65_LUX 0x1b37

Definition at line 53 of file usb.h.

Referenced by product_str().

9.42.2.38 #define P_K65_LUX_STR "1b37"

Definition at line 54 of file usb.h.

9.42.2.39 #define P_K65_NRGB 0x1b07

Definition at line 51 of file usb.h.

Referenced by product_str().

9.42.2.40 #define P_K65_NRGB_STR "1b07"

Definition at line 52 of file usb.h.

9.42.2.41 #define P_K65_RFIRE 0x1b39

Definition at line 55 of file usb.h.

Referenced by product_str().

9.42.2.42 #define P_K65_RFIRE_STR "1b39"

Definition at line 56 of file usb.h.

9.42.2.43 #define P_K65_STR "1b17"

Definition at line 50 of file usb.h.

9.42.2.44 #define P_K68 0x1b3f

Definition at line 59 of file usb.h.

Referenced by product_str().

9.42.2.45 #define P_K68_STR "1b3f"

Definition at line 60 of file usb.h.

9.42.2.46 #define P_K70 0x1b13

Definition at line 63 of file usb.h.

Referenced by product_str().

9.42.2.47 #define P_K70_LUX 0x1b33

Definition at line 67 of file usb.h.

Referenced by loadrgb_kb(), and product_str().

9.42.2.48 #define P_K70_LUX_NRGB 0x1b36

Definition at line 69 of file usb.h.

Referenced by loadrgb_kb(), and product_str().

9.42.2.49 #define P_K70_LUX_NRGB_STR "1b36"

Definition at line 70 of file usb.h.

9.42.2.50 #define P_K70_LUX_STR "1b33"

Definition at line 68 of file usb.h.

9.42.2.51 #define P_K70_NRGB 0x1b09

Definition at line 65 of file usb.h.

Referenced by product_str().

9.42.2.52 #define P_K70_NRGB_STR "1b09"

Definition at line 66 of file usb.h.

9.42.2.53 #define P_K70_RFIRE 0x1b38

Definition at line 71 of file usb.h.

Referenced by product_str().

9.42.2.54 #define P_K70_RFIRE_NRGB 0x1b3a

Definition at line 73 of file usb.h.

Referenced by product_str().

9.42.2.55 #define P_K70_RFIRE_NRGB_STR "1b3a"

Definition at line 74 of file usb.h.

9.42.2.56 #define P_K70_RFIRE_STR "1b38"

Definition at line 72 of file usb.h.

9.42.2.57 #define P_K70_STR "1b13"

Definition at line 64 of file usb.h.

9.42.2.58 #define P_K95 0x1b11

Definition at line 77 of file usb.h.

Referenced by product_str().

9.42.2.59 #define P_K95_NRGB 0x1b08

Definition at line 79 of file usb.h.

Referenced by _nk95cmd(), and product_str().

9.42.2.60 #define P_K95_NRGB_STR "1b08"

Definition at line 80 of file usb.h.

9.42.2.61 #define P_K95_PLATINUM 0x1b2d

Definition at line 81 of file usb.h.

Referenced by product_str(), and updatergb_kb().

9.42.2.62 #define P_K95_PLATINUM_STR "1b2d"

Definition at line 82 of file usb.h.

9.42.2.63 #define P_K95_STR "1b11"

Definition at line 78 of file usb.h.

9.42.2.64 #define P_M65 0x1b12

Definition at line 93 of file usb.h.

Referenced by product_str().

9.42.2.65 #define P_M65_PRO 0x1b2e

Definition at line 95 of file usb.h.

Referenced by product_str().

9.42.2.66 #define P_M65_PRO_STR "1b2e"

Definition at line 96 of file usb.h.

9.42.2.67 #define P_M65_STR "1b12"

Definition at line 94 of file usb.h.

9.42.2.68 #define P_SABRE_L 0x1b19 /* laser */

Definition at line 101 of file usb.h.

Referenced by product_str().

9.42.2.69 #define P_SABRE_L_STR "1b19"

Definition at line 102 of file usb.h.

9.42.2.70 #define P_SABRE_N 0x1b2f /* new? */

Definition at line 103 of file usb.h.

Referenced by product_str().

9.42.2.71 #define P_SABRE_N_STR "1b2f"

Definition at line 104 of file usb.h.

9.42.2.72 #define P_SABRE_O 0x1b14 /* optical */

Definition at line 99 of file usb.h.

Referenced by product_str().

9.42.2.73 #define P_SABRE_O2 0x1b32 /* Observed on a CH-9000111-EU model SABRE */

Definition at line 105 of file usb.h.

Referenced by product_str().

9.42.2.74 #define P_SABRE_O2_STR "1b32"

Definition at line 106 of file usb.h.

9.42.2.75 #define P_SABRE_O_STR "1b14"

Definition at line 100 of file usb.h.

9.42.2.76 #define P_SCIMITAR 0x1b1e

Definition at line 109 of file usb.h.

Referenced by product_str().

9.42.2.77 #define P_SCIMITAR_PRO 0x1b3e

Definition at line 111 of file usb.h.

Referenced by product_str().

9.42.2.78 #define P_SCIMITAR_PRO_STR "1b3e"

Definition at line 112 of file usb.h.

9.42.2.79 #define P_SCIMITAR_STR "1b1e"

Definition at line 110 of file usb.h.

9.42.2.80 #define P_STRAFE 0x1b20

Definition at line 85 of file usb.h.

Referenced by product_str().

9.42.2.81 #define P_STRAFE_NRGB 0x1b15

Definition at line 87 of file usb.h.

Referenced by product_str().

9.42.2.82 #define P_STRAFE_NRGB_2 0x1b44

Definition at line 89 of file usb.h.

Referenced by product_str().

9.42.2.83 #define P_STRAFE_NRGB_2_STR "1b44"

Definition at line 90 of file usb.h.

9.42.2.84 #define P_STRAFE_NRGB_STR "1b15"

Definition at line 88 of file usb.h.

9.42.2.85 #define P_STRAFE_STR "1b20"

Definition at line 86 of file usb.h.

9.42.2.86 #define resetusb(*kb*) _resetusb(kb, __FILE_NOPATH__, __LINE__)

Definition at line 246 of file usb.h.

Referenced by usb_tryreset().

9.42.2.87 #define USB_DELAY_DEFAULT 5

Definition at line 192 of file usb.h.

Referenced by _setupusb(), and start_dev().

9.42.2.88 #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 288 of file usb.h.

Referenced by cmd_hwload_kb(), cmd_hwload_mouse(), getfwversion(), hwloadmode(), loaddpi(), loadrgb_kb(), and loadrgb_mouse().

9.42.2.89 #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 271 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().

9.42.2.90 #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 42 of file usb.h.

Referenced by usb_add_device(), and vendor_str().

9.42.2.91 #define V_CORSAIR_STR "1b1c"

Definition at line 43 of file usb.h.

Referenced by udev_enum(), and usb_add_device().

9.42.3 Function Documentation

9.42.3.1 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 189 of file usb_linux.c.

References ckb_err_fn, usbdevice::handle, P_K95_NRGB, and usbdevice::product.

```
189                                                                                 {
190     if(kb->product != P_K95_NRGB)
191         return 0;
192     struct usbdevfs_ctrltransfer transfer = { 0x40, bRequest, wValue, 0, 0, 5000, 0 };
193     int res = ioctl(kb->handle - 1, USBDEVFS_CONTROL, &transfer);
194     if(res <= 0){
195         ckb_err_fn("%s\n", file, line, res ? strerror(errno) : "No data written");
196         return 1;
197     }
198     return 0;
199 }
```

9.42.3.2 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.

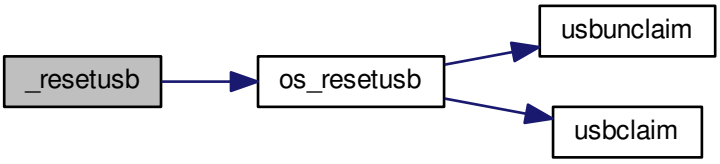
First reset the device via os_resetusb() after a long delay (it may send something to the host). If this worked (retval == 0), give the device another long delay Then perform the initialization via the device specific start() function entry in kb->vtable and if this is successful also, return the result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

Definition at line 436 of file usb.c.

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

```
436                                                     {
437     // Perform a USB reset
438     DELAY_LONG(kb);
439     int res = os_resetusb(kb, file, line);
440     if(res)
441         return res;
442     DELAY_LONG(kb);
443     // Re-initialize the device
444     if(kb->vtable->start(kb, kb->active) != 0)
445         return -1;
446     if(kb->vtable->updatergb(kb, 1) != 0)
447         return -1;
448     return 0;
449 }
```

Here is the call graph for this function:



9.42.3.3 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. < Synchonization between macro and color information

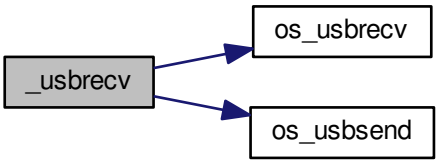
Definition at line 611 of file usb.c.

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

```
611                                                                                                 {
612     // Try a maximum of 5 times
613     for (int try = 0; try < 5; try++) {
614         // Send the output message
615         pthread_mutex_lock(mmutex(kb));
616         DELAY_SHORT(kb);
617         int res = os_usbsend(kb, out_msg, 1, file, line);
618         pthread_mutex_unlock(mmutex(kb));
619         if (res == 0)
620             return 0;
621         else if (res == -1) {
622             // Retry on temporary failure
623             if (reset_stop)
624                 return 0;
625             DELAY_LONG(kb);
626             continue;
627         }
```

```
628         // Wait for the response
629         DELAY_MEDIUM(kb);
630         res = os_usbrecv(kb, in_msg, file, line);
631         if(res == 0)
632             return 0;
633         else if(res != -1)
634             return res;
635         if(reset_stop || hwload_mode != 2)
636             return 0;
637         DELAY_LONG(kb);
638     }
639     // Give up
640     ckb_err_fn("Too many send/recv failures. Dropping.\n", file, line);
641     return 0;
642 }
```

Here is the call graph for this function:



9.42.3.4 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 (...). < Synchonization between macro and color information

Definition at line 542 of file usb.c.

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

```
542                                                                                                 {
543     int total_sent = 0;
544     for(int i = 0; i < count; i++){
545         // Send each message via the OS function
546         while(1){
547             pthread_mutex_lock(mmutex(kb));
548             DELAY_SHORT(kb);
549             int res = os_usbsend(kb, messages + i * MSG_SIZE, 0, file, line);
550             pthread_mutex_unlock(mmutex(kb));
551             if(res == 0)
552                 return 0;
553             else if(res != -1){
554                 total_sent += res;
555                 break;
556             }
557             // Stop immediately if the program is shutting down or hardware load is set to tryonce
558             if(reset_stop || hwload_mode != 2)
559                 return 0;
560             // Retry as long as the result is temporary failure
561             DELAY_LONG(kb);
562         }
563     }
564     return total_sent;
565 }
```

Here is the call graph for this function:

9.42.3.5 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).

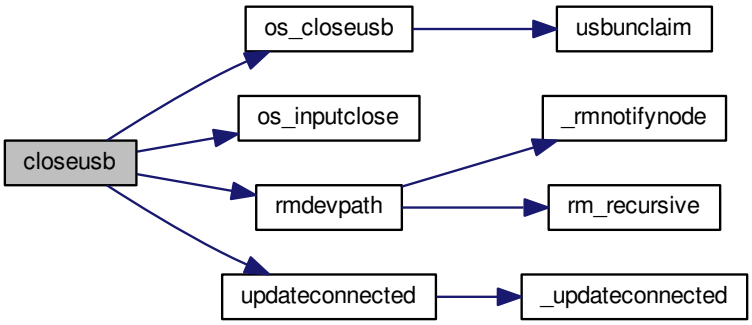
Definition at line 687 of file usb.c.

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

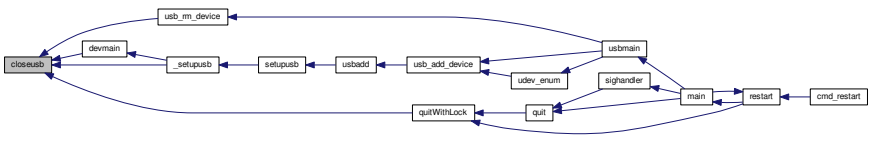
Referenced by _setupusb(), devmain(), quitWithLock(), and usb_rm_device().

```
687                             {
688     pthread_mutex_lock(imutex(kb));
689     if(kb->handle){
690         int index = INDEX_OF(kb, keyboard);
691         ckb_info("Disconnecting %s%d\n", devpath, index);
692         os_inputclose(kb);
693         updateconnected();
694         // Close USB device
695         os_closeusb(kb);
696     } else
697         updateconnected();
698     rmdevpath(kb);
699
700     // Wait for thread to close
701     pthread_mutex_unlock(imutex(kb));
702     pthread_mutex_unlock(dmutex(kb));
703     pthread_join(kb->thread, 0);
704     pthread_mutex_lock(dmutex(kb));
705
706     // Delete the profile and the control path
707     if(!kb->vtable)
708         return 0;
709     kb->vtable->freeprofile(kb);
710     memset(kb, 0, sizeof(usbdevice));
711     return 0;
712 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



9.42.3.6 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 448 of file usb_linux.c.

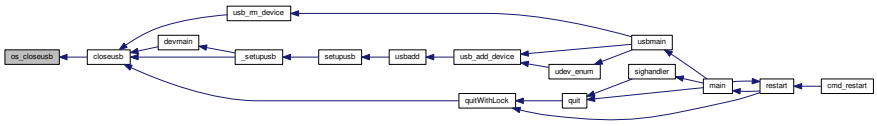
References usbdevice::handle, INDEX_OF, kbsyspath, keyboard, usbdevice::udev, and usbunclaim().

Referenced by closeusb().

```
448                                {
449     if(kb->handle){
450         usbunclaim(kb, 0);
451         close(kb->handle - 1);
452     }
453     if(kb->udev)
454         udev_device_unref(kb->udev);
455     kb->handle = 0;
456     kb->udev = 0;
457     kbsyspath[INDEX_OF(kb, keyboard)][0] = 0;
458 }
```

Here is the call graph for this function:

Here is the caller graph for this function:



9.42.3.7 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, 64 for FW v3
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 248 of file usb_linux.c.

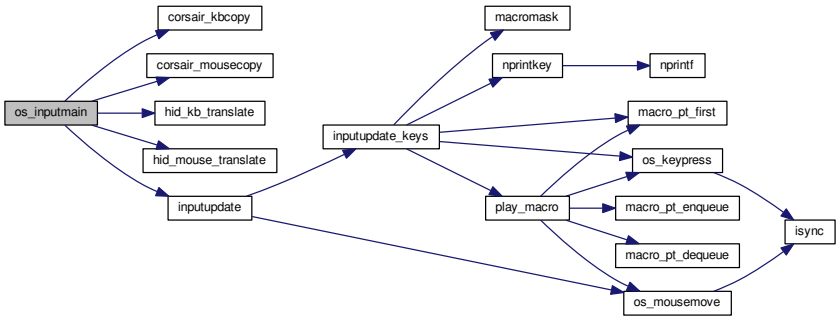
References usbdevice::active, ckb_err, ckb_info, corsair_kbcopy(), corsair_mousecopy(), devpath, usbdevice::epcount, usbdevice::fwversion, usbdevice::handle, hid_kb_translate(), hid_mouse_translate(), imutex, INDEX_OF, usbdevice::input, inputupdate(), IS_MOUSE, IS_RGB, keyboard, usbinput::keys, MSG_SIZE, usbdevice::product, usbinput::rel_x, usbinput::rel_y, and usbdevice::vendor.

Referenced by _setupusb().

```
248                                 {
249     usbdevice* kb = context;
250     int fd = kb->handle - 1;
251     short vendor = kb->vendor, product = kb->product;
252     int index = INDEX_OF(kb, keyboard);
253     ckb_info("Starting input thread for %s%d\n", devpath, index);
254 
259     int urbcount = IS_RGB(vendor, product) ? (kb->epcount - 1) : kb->
    epcount;
260     if (urbcount == 0) {
261         ckb_err("urbcount = 0, so there is nothing to claim in os_inputmain()\n");
262         return 0;
263     }
264 
266     struct usbdevfs_urb urbs[urbcount + 1];
267     memset(urbs, 0, sizeof(urbs));
268 
282     urbs[0].buffer_length = (kb->fwversion >= 0x300 ? MSG_SIZE : 8);
283     if(urbcount > 1 && IS_RGB(vendor, product)) {
284         if(IS_MOUSE(vendor, product))
285             urbs[1].buffer_length = 10;
286         else
287             urbs[1].buffer_length = 21;
288         urbs[2].buffer_length = MSG_SIZE;
289         if(urbcount != 3)
290             urbs[urbcount - 1].buffer_length = MSG_SIZE;
291     } else if(kb->fwversion < 0x300) {
292             urbs[1].buffer_length = 4;
293             urbs[2].buffer_length = 15;
294     }
295 
298     for(int i = 0; i < urbcount; i++){
299         urbs[i].type = USBDEVFS_URB_TYPE_INTERRUPT;
300         urbs[i].endpoint = 0x80 | (i + 1);
301         urbs[i].buffer = malloc(urbs[i].buffer_length);
302         ioctl(fd, USBDEVFS_SUBMITURB, urbs + i);
303     }
304 
306     while (1) {
307         struct usbdevfs_urb* urb = 0;
308 
311         if (ioctl(fd, USBDEVFS_REAPURB, &urb)) {
312             if (errno == ENODEV || errno == ENOENT || errno == ESHUTDOWN)
313                 // Stop the thread if the handle closes
314                 break;
315             else if(errno == EPIPE && urb){
317                 ioctl(fd, USBDEVFS_CLEAR_HALT, &urb->endpoint);
318                 // Re-submit the URB
319                 if(urb)
```

```
320                     ioctl(fd, USBDEVFS_SUBMITURB, urb);
321                 urb = 0;
322             }
323             continue;
324         }
325 
329         if (urb) {
330 
342             pthread_mutex_lock(imutex(kb));
343             // EP workaround for FWv3
344             // Corsair input comes through 0x81, but case 1 in keymap.c is used for 6KRO
345             uchar urbendpoint = (kb->fwversion >= 0x300 ? 2 : (urb->endpoint & 0xF));
346             if(IS_MOUSE(vendor, product)){
347                 switch(urb->actual_length){
348                 case 8:
349                 case 10:
350                 case 11:
351                     // HID mouse input
352                     hid_mouse_translate(kb->input.keys, &kb->
    input.rel_x, &kb->input.rel_y, -urbendpoint, urb->actual_length, urb->buffer, kb->
    fwversion);
353                     break;
354                 case MSG_SIZE:
355                     // Corsair mouse input
356                     corsair_mousecopy(kb->input.keys, -urbendpoint, urb->buffer);
357                     break;
358                 }
359             } else if(IS_RGB(vendor, product)){
360                 switch(urb->actual_length){
361                 case 8:
362                     // RGB EP 1: 6KRO (BIOS mode) input
363                     hid_kb_translate(kb->input.keys, -1, urb->actual_length, urb->
    buffer);
364                     break;
365                 case 21:
366                 case 5:
367                     // RGB EP 2: NKRO (non-BIOS) input. Accept only if keyboard is inactive
368                     if(!kb->active)
369                         hid_kb_translate(kb->input.keys, -2, urb->actual_length,
    urb->buffer);
370                     break;
371                 case MSG_SIZE:
372                     // RGB EP 3: Corsair input
373                     corsair_kbcopy(kb->input.keys, -urbendpoint, urb->buffer);
374                     break;
375                 }
376             } else {
377                 // Non-RGB input
378                 hid_kb_translate(kb->input.keys, urb->endpoint & 0xF, urb->
    actual_length, urb->buffer);
379             }
382             inputupdate(kb);
383             pthread_mutex_unlock(imutex(kb));
384 
386             ioctl(fd, USBDEVFS_SUBMITURB, urb);
387             urb = 0;
388         }
389     }
390 
394     ckb_info("Stopping input thread for %s%d\n", devpath, index);
395     for(int i = 0; i < urbcount; i++){
396         ioctl(fd, USBDEVFS_DISCARDURB, urbs + i);
397         free(urbs[i].buffer);
398     }
399     return 0;
400 }
```

Here is the call graph for this function:



Here is the caller graph for this function:

9.42.3.8 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...

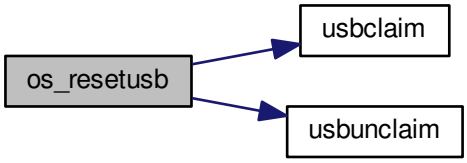
Definition at line 510 of file usb_linux.c.

References usbdevice::handle, TEST_RESET, usbclaim(), and usbunclaim().

Referenced by _resetusb().

```
510                                                     {
511     TEST_RESET(usbunclaim(kb, 1));
512     TEST_RESET(ioctl(kb->handle - 1, USBDEVFS_RESET));
513     TEST_RESET(usbclaim(kb));
514     // Success!
515     return 0;
516 }
```

Here is the call graph for this function:



Here is the caller graph for this function:

9.42.3.9 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 214 of file usb_linux.c.

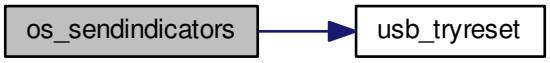
References ckb_err, usbdevice::fwversion, usbdevice::handle, usbdevice::ileds, and usb_tryreset().

Referenced by updateindicators_kb().

```
214                                     {
```

```
215     static int countForReset = 0;
216     void *ileds;
217     ushort leds;
218     if(kb->fwversion >= 0x300) {
219         leds = (kb->ileds << 8) | 0x0001;
220         ileds = &leds;
221     }
222     else {
223         ileds = &kb->ileds;
224     }
225     struct usbdevfs_ctrltransfer transfer = { 0x21, 0x09, 0x0200, 0x00, (kb->
      fwversion >= 0x300 ? 2 : 1), 500, ileds };
226     int res = ioctl(kb->handle - 1, USBDEVFS_CONTROL, &transfer);
227     if(res <= 0) {
228         ckb_err("%s\n", res ? strerror(errno) : "No data written");
229         if (usb_tryreset(kb) == 0 && countForReset++ < 3) {
230             os_sendindicators(kb);
231         }
232     }
233 }
```

Here is the call graph for this function:



Here is the caller graph for this function:

9.42.3.10 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 about connecting interfaces
- Claim the USB interfaces

Todo in these modules a pullrequest is outstanding

< Try to reset the device and recall the function

< Don't do this endless in recursion

< os_setupusb() has a return value (used as boolean)

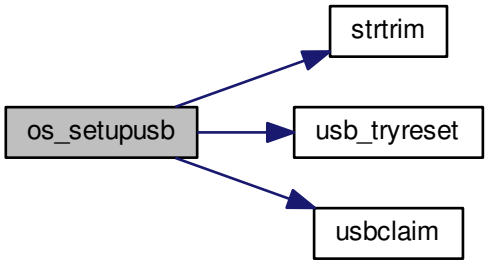
Definition at line 548 of file usb_linux.c.

References ckb_err, ckb_info, devpath, usbdevice::epcount, usbdevice::fwversion, INDEX_OF, KB_NAME_LEN, keyboard, usbdevice::name, usbdevice::serial, SERIAL_LEN, strtrim(), usbdevice::udev, usb_tryreset(), and usbclaim().

Referenced by _setupusb().

```
548                                {
551     struct udev_device* dev = kb->udev;
552     const char* name = udev_device_get_sysattr_value(dev, "product");
553     if(name)
554         strncpy(kb->name, name, KB_NAME_LEN);
555     strtrim(kb->name);
556     const char* serial = udev_device_get_sysattr_value(dev, "serial");
557     if(serial)
558         strncpy(kb->serial, serial, SERIAL_LEN);
559     strtrim(kb->serial);
562     const char* firmware = udev_device_get_sysattr_value(dev, "bcdDevice");
563     if(firmware)
564         sscanf(firmware, "%hx", &kb->fwversion);
565     else
566         kb->fwversion = 0;
567     int index = INDEX_OF(kb, keyboard);
568
570     ckb_info("Connecting %s at %s%d\n", kb->name, devpath, index);
571
577     const char* ep_str = udev_device_get_sysattr_value(dev, "bNumInterfaces");
578 #ifdef DEBUG
579     ckb_info("claiming interfaces. name=%s, firmware=%s; ep_str=%s\n", name, firmware, ep_str);
580 #endif //DEBUG
581     kb->epcount = 0;
582     if(ep_str)
583         sscanf(ep_str, "%d", &kb->epcount);
584     if(kb->epcount < 2){
585         // IF we have an RGB KB with 0 or 1 endpoints, it will be in BIOS mode.
586         ckb_err("Unable to read endpoint count from udev, assuming %d and reading >>%s<< or device
      is in BIOS mode\n", kb->epcount, ep_str);
587         if (usb_tryreset(kb) == 0) {
588             static int retryCount = 0;
589             if (retryCount++ < 5) {
590                 return os_setupusb(kb);
591             }
592         }
593         return -1;
594         // ToDo are there special versions we have to detect? If there are, that was the old code to handle
      it:
595         // This shouldn't happen, but if it does, assume EP count based onckb_warn what the device is
      supposed to have
596         // kb->epcount = (HAS_FEATURES(kb, FEAT_RGB) ? 4 : 3);
597         // ckb_warn("Unable to read endpoint count from udev, assuming %d and reading >>%s<<...\n",
      kb->epcount, ep_str);
598     }
599     if(usbclaim(kb)){
600         ckb_err("Failed to claim interfaces: %s\n", strerror(errno));
601         return -1;
602     }
603     return 0;
604 }
```

Here is the call graph for this function:



Here is the caller graph for this function:

9.42.3.11 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, Recipient=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 130 of file usb_linux.c.

References ckb_err_fn, ckb_warn_fn, usbdevice::epcount, usbdevice::handle, and MSG_SIZE.

Referenced by _usbrecv().

```
130                                                                         {
131     int res;
132     // This is what CUE does, but it doesn't seem to work on linux.
133     /*if(kb->fwversion >= 0x130){
134         struct usbdevfs_bulktransfer transfer = {0};
135         transfer.ep = 0x84;
136         transfer.len = MSG_SIZE;
137         transfer.timeout = 5000;
138         transfer.data = in_msg;
139         res = ioctl(kb->handle - 1, USBDEVFS_BULK, &transfer);
140     } else {*/
141         struct usbdevfs_ctrltransfer transfer = { 0xa1, 0x01, 0x0300, kb->
    epcount - 1, MSG_SIZE, 5000, in_msg };
142         res = ioctl(kb->handle - 1, USBDEVFS_CONTROL, &transfer);
143     //}
144     if(res <= 0){
145         ckb_err_fn("%s\n", file, line, res ? strerror(errno) : "No data read");
146         if(res == -1 && errno == ETIMEDOUT)
147             return -1;
148         else
149             return 0;
150     } else if(res != MSG_SIZE)
151         ckb_warn_fn("Read %d bytes (expected %d)\n", file, line, res,
    MSG_SIZE);
152 #ifdef DEBUG_USB_RECV
153     char converted[MSG_SIZE*3 + 1];
154     for(int i=0;i<MSG_SIZE;i++)
155         sprintf(&converted[i*3], "%02x ", in_msg[i]);
156     ckb_warn_fn("Recv %s\n", file, line, converted);
157 #endif
158     return res;
159 }
```

Here is the caller graph for this function:

9.42.3.12 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, Recipient=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_SEND is set during compilation, the number of bytes sent and their representation are logged to the error channel.

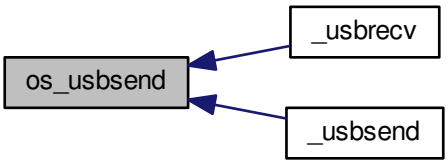
Definition at line 68 of file usb_linux.c.

References ckb_err_fn, ckb_warn_fn, usbdevice::epcount, usbdevice::fwversion, usbdevice::handle, IS_V2_OVERRIDE, and MSG_SIZE.

Referenced by _usbrecv(), and _usbsend().

```
68                                                                                                  {
69     int res;
70     if ((kb->fwversion >= 0x120 || IS_V2_OVERRIDE(kb)) && !is_recv){
71         struct usbdevfs_bulktransfer transfer = {0};
72         // FW 2.XX uses 0x03, FW 3.XX uses 0x02
73         transfer.ep = (kb->fwversion >= 0x130 && kb->fwversion < 0x200) ? 4 : (kb->
       fwversion >= 0x300 ? 2 : 3);
74         transfer.len = MSG_SIZE;
75         transfer.timeout = 5000;
76         transfer.data = (void*)out_msg;
77         res = ioctl(kb->handle - 1, USBDEVFS_BULK, &transfer);
78     } else {
79         struct usbdevfs_ctrltransfer transfer = { 0x21, 0x09, 0x0200, kb->
       epcount - 1, MSG_SIZE, 5000, (void*)out_msg };
80         res = ioctl(kb->handle - 1, USBDEVFS_CONTROL, &transfer);
81     }
82
83     if (res <= 0){
84         ckb_err_fn(" %s, res = 0x%x\n", file, line, res ? strerror(errno) : "No data written",
       res);
85         if(res == -1 && errno == ETIMEDOUT)
86             return -1;
87         else
88             return 0;
89     } else if (res != MSG_SIZE)
90         ckb_warn_fn("Wrote %d bytes (expected %d)\n", file, line, res,
       MSG_SIZE);
91 #ifdef DEBUG_USB_SEND
92     char converted[MSG_SIZE*3 + 1];
93     for(int i=0;i<MSG_SIZE;i++)
94         sprintf(&converted[i*3], "%02x ", out_msg[i]);
95     ckb_warn_fn("Sent %s\n", file, line, converted);
96 #endif
97     return res;
98 }
```

Here is the caller graph for this function:



9.42.3.13 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*

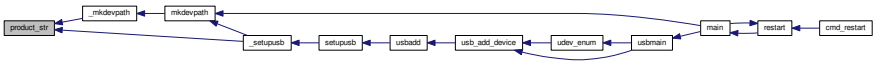
Definition at line 70 of file usb.c.

References P_GLAIVE, P_HARPOON, P_K63_NRGB, P_K65, P_K65_LUX, P_K65_NRGB, P_K65_RFIRE, P-_K68, P_K70, P_K70_LUX, P_K70_LUX_NRGB, P_K70_NRGB, P_K70_RFIRE, P_K70_RFIRE_NRGB, P_K95, P_K95_NRGB, P_K95_PLATINUM, P_M65, P_M65_PRO, P_SABRE_L, P_SABRE_N, P_SABRE_O, P_SABRE-_O2, P_SCIMITAR, P_SCIMITAR_PRO, P_STRAFE, P_STRAFE_NRGB, and P_STRAFE_NRGB_2.

Referenced by _mkdevpath(), and _setupusb().

```
70                                      {
71     if(product == P_K95 || product == P_K95_NRGB)
72         return "k95";
73     if(product == P_K95_PLATINUM)
74         return "k95p";
75     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)
76         return "k70";
77     if(product == P_K68)
78         return "k68";
79     if(product == P_K65 || product == P_K65_NRGB || product ==
       P_K65_LUX || product == P_K65_RFIRE)
80         return "k65";
81     if(product == P_K63_NRGB)
82         return "k63";
83     if(product == P_STRAFE || product == P_STRAFE_NRGB || product ==
       P_STRAFE_NRGB_2)
84         return "strafe";
85     if(product == P_M65 || product == P_M65_PRO)
86         return "m65";
87     if(product == P_SABRE_O || product == P_SABRE_L || product ==
       P_SABRE_N || product == P_SABRE_O2)
88         return "sabre";
89     if(product == P_SCIMITAR || product == P_SCIMITAR_PRO)
90         return "scimitar";
91     if(product == P_HARPOON)
92         return "harpoon";
93     if(product == P_GLAIVE)
94         return "glaive";
95     return "";
96 }
```

Here is the caller graph for this function:



9.42.3.14 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 417 of file usb.c.

References FEAT_RGB, HAS_FEATURES, NEEDS_FW_UPDATE, NK95_HWON, nk95cmd, and setactive.

Referenced by quitWithLock().

```
417                             {
418     if(NEEDS_FW_UPDATE(kb))
419         return 0;
420     if(!HAS_FEATURES(kb, FEAT_RGB)){
421         nk95cmd(kb, NK95_HWON);
422         return 0;
423     }
424     if(setactive(kb, 0))
425         return -1;
426     return 0;
427 }
```

Here is the caller graph for this function:

9.42.3.15 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.

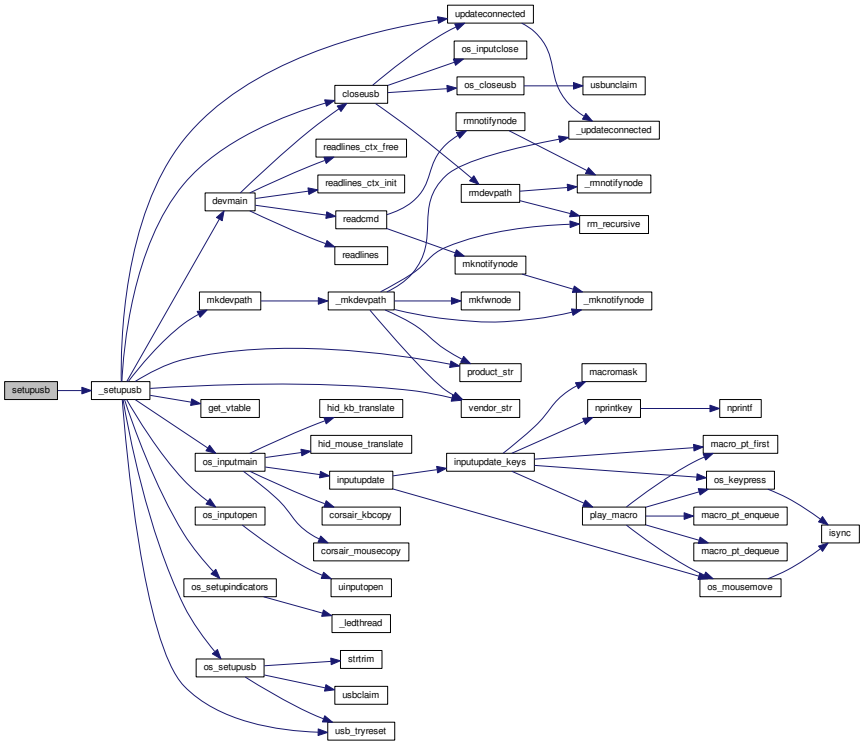
Definition at line 396 of file usb.c.

References _setupusb(), ckb_err, imutex, and usbdevice::thread.

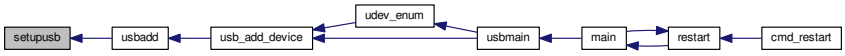
Referenced by usbadd().

```
396                             {
397     pthread_mutex_lock(imutex(kb));
398     if(pthread_create(&kb->thread, 0, _setupusb, kb))
399         ckb_err("Failed to create USB thread\n");
400 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



9.42.3.16 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?

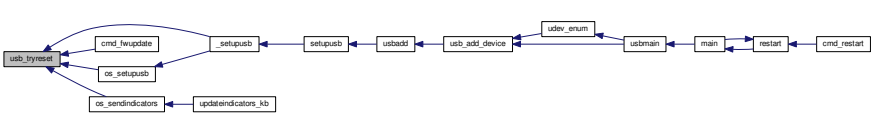
Definition at line 475 of file usb.c.

References ckb_err, ckb_info, reset_stop, and resetusb.

Referenced by _setupusb(), cmd_fwupdate(), os_sendindicators(), and os_setupusb().

```
475                                 {
476     if(reset_stop)
477         return -1;
478     ckb_info("Attempting reset...\n");
479     while(1){
480         int res = resetusb(kb);
481         if(!res){
482             ckb_info("Reset success\n");
483             return 0;
484         }
485         if(res == -2 || reset_stop)
486             break;
487     }
488     ckb_err("Reset failed. Disconnecting.\n");
489     return -1;
490 }
```

Here is the caller graph for this function:



9.42.3.17 void usbkill ( )

Definition at line 853 of file usb_linux.c.

Referenced by quitWithLock().

```
853             {
854     udev_unref(udev);
855     udev = 0;
856 }
```

Here is the caller graph for this function:

9.42.3.18 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 793 of file usb_linux.c.

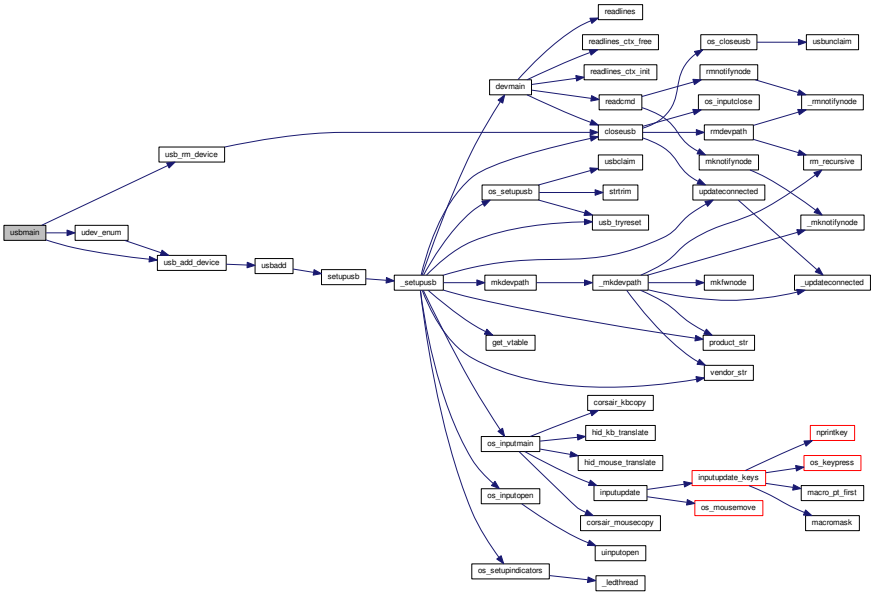
References ckb_fatal, ckb_warn, udev_enum(), usb_add_device(), and usb_rm_device().

Referenced by main().

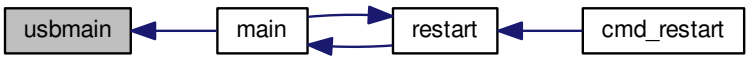
```
793             {
798     // Load the uinput module (if it's not loaded already)
799     if(system("modprobe uinput") != 0)
800         ckb_warn("Failed to load uinput module\n");
801
805     if(!(udev = udev_new())) {
806         ckb_fatal("Failed to initialize udev in usbmain(), usb_linux.c\n");
807         return -1;
808     }
809
812     udev_enum();
813
816     // Done scanning. Enter a loop to poll for device updates
817     struct udev_monitor* monitor = udev_monitor_new_from_netlink(udev, "udev");
818     udev_monitor_filter_add_match_subsystem_devtype(monitor, "usb", 0);
819     udev_monitor_enable_receiving(monitor);
820     // Get an fd for the monitor
821     int fd = udev_monitor_get_fd(monitor);
822     fd_set fds;
823     while(udev){
824         FD_ZERO(&fds);
825         FD_SET(fd, &fds);
826         // Block until an event is read
827         if(select(fd + 1, &fds, 0, 0, 0) > 0 && FD_ISSET(fd, &fds)){
828             struct udev_device* dev = udev_monitor_receive_device(monitor);
829             if(!dev)
830                 continue;
```

```
831             const char* action = udev_device_get_action(dev);
832             if(!action){
833                 udev_device_unref(dev);
834                 continue;
835             }
836             // Add/remove device
837             if(!strcmp(action, "add")){
838                 int res = usb_add_device(dev);
839                 if(res == 0)
840                     continue;
841                 // If the device matched but the handle wasn't opened correctly, re-enumerate (this
      sometimes solves the problem)
842                 if(res == -1)
843                     udev_enum();
844             } else if(!strcmp(action, "remove"))
845                 usb_rm_device(dev);
846             udev_device_unref(dev);
847         }
848     }
849     udev_monitor_unref(monitor);
850     return 0;
851 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



9.42.3.19 const char* vendor_str (short *vendor*)

uncomment to see USB packets received from the device vendor_str Vendor/product string representations

Parameters

vendor	*short* vendor ID

Returns

a string: either "" or "corsair"

uncomment to see USB packets sent to the device

vendor_str returns "corsair" if 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".

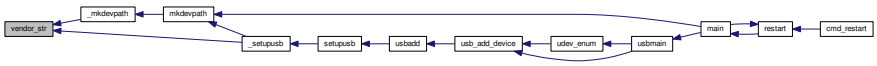
Definition at line 43 of file usb.c.

References V_CORSAIR.

Referenced by _mkdevpath(), and _setupusb().

```
43                                      {
44     if(vendor == V_CORSAIR)
45         return "corsair";
46     return "";
47 }
```

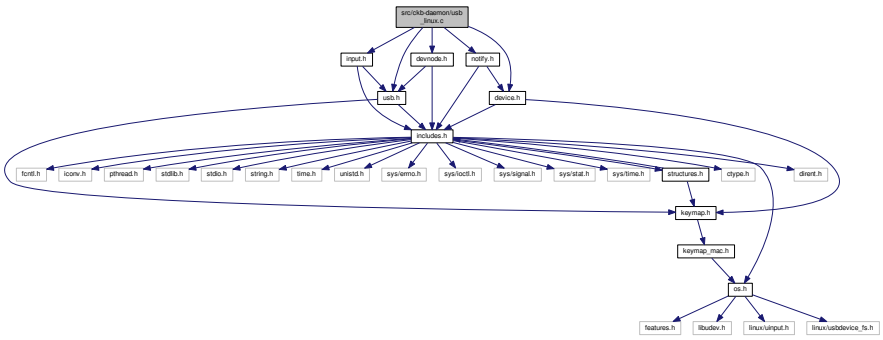
Here is the caller graph for this function:



9.43 src/ckb-daemon/usb_linux.c File Reference

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

Include dependency graph for usb_linux.c:



Data Structures

- struct _model

Macros

- #define DEBUG

  all open usb devices have their system path names here in this array.
- #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]
- static struct udev * udev

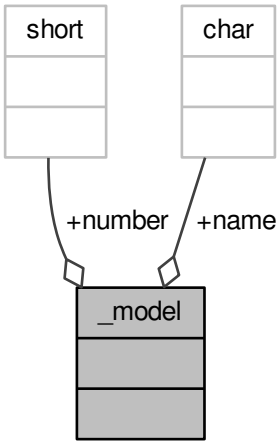
  struct udef is defined in /usr/include/libudev.h
- pthread_t usbthread
- pthread_t udevthread
- static _model models []

9.43.1 Data Structure Documentation

9.43.1.1 struct _model

Definition at line 658 of file usb_linux.c.

Collaboration diagram for _model:



Data Fields

const char *	name	
short	number	

9.43.2 Macro Definition Documentation

9.43.2.1 #define DEBUG

Definition at line 11 of file usb_linux.c.

9.43.2.2 #define N_MODELS (sizeof(models) / sizeof(_model))

Definition at line 700 of file usb_linux.c.

Referenced by usb_add_device().

9.43.2.3 #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 492 of file usb_linux.c.

Referenced by os_resetusb().

9.43.3 Function Documentation

9.43.3.1 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, Recipient=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 189 of file usb_linux.c.

References ckb_err_fn, usbdevice::handle, P_K95_NRGB, and usbdevice::product.

```
189                                                                                 {
190     if(kb->product != P_K95_NRGB)
```

```
191         return 0;
192     struct usbdevfs_ctrltransfer transfer = { 0x40, bRequest, wValue, 0, 0, 5000, 0 };
193     int res = ioctl(kb->handle - 1, USBDEVFS_CONTROL, &transfer);
194     if(res <= 0){
195         ckb_err_fn("%s\n", file, line, res ? strerror(errno) : "No data written");
196         return 1;
197     }
198     return 0;
199 }
```

9.43.3.2 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 448 of file usb_linux.c.

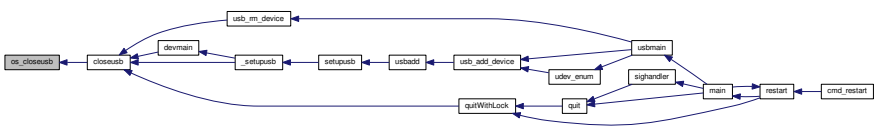
References usbdevice::handle, INDEX_OF, kbsyspath, keyboard, usbdevice::udev, and usbunclaim().

Referenced by closeusb().

```
448                                 {
449     if(kb->handle){
450         usbunclaim(kb, 0);
451         close(kb->handle - 1);
452     }
453     if(kb->udev)
454         udev_device_unref(kb->udev);
455     kb->handle = 0;
456     kb->udev = 0;
457     kbsyspath[INDEX_OF(kb, keyboard)][0] = 0;
458 }
```

Here is the call graph for this function:

Here is the caller graph for this function:



9.43.3.3 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, 64 for FW v3
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 248 of file usb_linux.c.

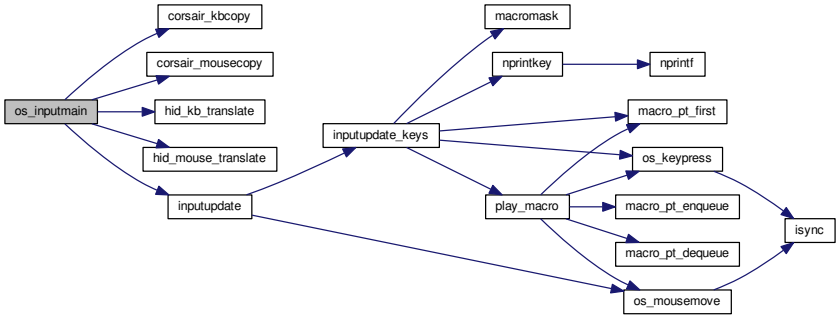
References usbdevice::active, ckb_err, ckb_info, corsair_kbcopy(), corsair_mousecopy(), devpath, usbdevice::epcount, usbdevice::fwversion, usbdevice::handle, hid_kb_translate(), hid_mouse_translate(), imutex, INDEX_OF, usbdevice::input, inputupdate(), IS_MOUSE, IS_RGB, keyboard, usbinput::keys, MSG_SIZE, usbdevice::product, usbinput::rel_x, usbinput::rel_y, and usbdevice::vendor.

Referenced by _setupusb().

```
248                                 {
249     usbdevice* kb = context;
250     int fd = kb->handle - 1;
251     short vendor = kb->vendor, product = kb->product;
252     int index = INDEX_OF(kb, keyboard);
253     ckb_info("Starting input thread for %s%d\n", devpath, index);
254 
259     int urbcount = IS_RGB(vendor, product) ? (kb->epcount - 1) : kb->
    epcount;
260     if (urbcount == 0) {
261         ckb_err("urbcount = 0, so there is nothing to claim in os_inputmain()\n");
262         return 0;
263     }
264 
266     struct usbdevfs_urb urbs[urbcount + 1];
267     memset(urbs, 0, sizeof(urbs));
268 
282     urbs[0].buffer_length = (kb->fwversion >= 0x300 ? MSG_SIZE : 8);
283     if(urbcount > 1 && IS_RGB(vendor, product)) {
284         if(IS_MOUSE(vendor, product))
285             urbs[1].buffer_length = 10;
286         else
287             urbs[1].buffer_length = 21;
288         urbs[2].buffer_length = MSG_SIZE;
289         if(urbcount != 3)
290             urbs[urbcount - 1].buffer_length = MSG_SIZE;
291     } else if(kb->fwversion < 0x300) {
292             urbs[1].buffer_length = 4;
293             urbs[2].buffer_length = 15;
294     }
295 
298     for(int i = 0; i < urbcount; i++){
299         urbs[i].type = USBDEVFS_URB_TYPE_INTERRUPT;
300         urbs[i].endpoint = 0x80 | (i + 1);
301         urbs[i].buffer = malloc(urbs[i].buffer_length);
302         ioctl(fd, USBDEVFS_SUBMITURB, urbs + i);
303     }
304 
306     while (1) {
307         struct usbdevfs_urb* urb = 0;
308 
311         if (ioctl(fd, USBDEVFS_REAPURB, &urb)) {
312             if (errno == ENODEV || errno == ENOENT || errno == ESHUTDOWN)
313                 // Stop the thread if the handle closes
314                 break;
315             else if(errno == EPIPE && urb){
317                 ioctl(fd, USBDEVFS_CLEAR_HALT, &urb->endpoint);
318                 // Re-submit the URB
319                 if(urb)
320                     ioctl(fd, USBDEVFS_SUBMITURB, urb);
321                 urb = 0;
322             }
323             continue;
324         }
325 
329         if (urb) {
330 
342             pthread_mutex_lock(imutex(kb));
```

```
343             // EP workaround for FWv3
344             // Corsair input comes through 0x81, but case 1 in keymap.c is used for 6KRO
345             uchar urbendpoint = (kb->fwversion >= 0x300 ? 2 : (urb->endpoint & 0xF));
346             if(IS_MOUSE(vendor, product)){
347                 switch(urb->actual_length){
348                 case 8:
349                 case 10:
350                 case 11:
351                     // HID mouse input
352                     hid_mouse_translate(kb->input.keys, &kb->
    input.rel_x, &kb->input.rel_y, -urbendpoint, urb->actual_length, urb->buffer, kb->
    fwversion);
353                     break;
354                 case MSG_SIZE:
355                     // Corsair mouse input
356                     corsair_mousecopy(kb->input.keys, -urbendpoint, urb->buffer);
357                     break;
358                 }
359             } else if(IS_RGB(vendor, product)){
360                 switch(urb->actual_length){
361                 case 8:
362                     // RGB EP 1: 6KRO (BIOS mode) input
363                     hid_kb_translate(kb->input.keys, -1, urb->actual_length, urb->
    buffer);
364                     break;
365                 case 21:
366                 case 5:
367                     // RGB EP 2: NKRO (non-BIOS) input. Accept only if keyboard is inactive
368                     if(!kb->active)
369                         hid_kb_translate(kb->input.keys, -2, urb->actual_length,
    urb->buffer);
370                     break;
371                 case MSG_SIZE:
372                     // RGB EP 3: Corsair input
373                     corsair_kbcopy(kb->input.keys, -urbendpoint, urb->buffer);
374                     break;
375                 }
376             } else {
377                 // Non-RGB input
378                 hid_kb_translate(kb->input.keys, urb->endpoint & 0xF, urb->
    actual_length, urb->buffer);
379             }
382             inputupdate(kb);
383             pthread_mutex_unlock(imutex(kb));
384
386             ioctl(fd, USBDEVFS_SUBMITURB, urb);
387             urb = 0;
388         }
389     }
390
394     ckb_info("Stopping input thread for %s%d\n", devpath, index);
395     for(int i = 0; i < urbcount; i++){
396         ioctl(fd, USBDEVFS_DISCARDURB, urbs + i);
397         free(urbs[i].buffer);
398     }
399     return 0;
400 }
```

Here is the call graph for this function:



Here is the caller graph for this function:

9.43.3.4 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...

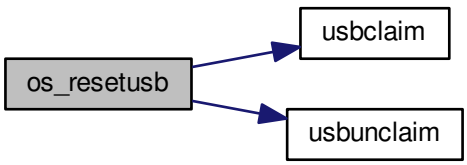
Definition at line 510 of file usb_linux.c.

References usbdevice::handle, TEST_RESET, usbclaim(), and usbunclaim().

Referenced by _resetusb().

```
510                                                         {
511     TEST_RESET(usbunclaim(kb, 1));
512     TEST_RESET(ioctl(kb->handle - 1, USBDEVFS_RESET));
513     TEST_RESET(usbclaim(kb));
514     // Success!
515     return 0;
516 }
```

Here is the call graph for this function:



Here is the caller graph for this function:

9.43.3.5 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.

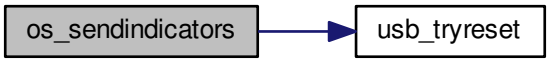
Definition at line 214 of file usb_linux.c.

References ckb_err, usbdevice::fwversion, usbdevice::handle, usbdevice::ileds, and usb_tryreset().

Referenced by updateindicators_kb().

```
214                                     {
215     static int countForReset = 0;
216     void *ileds;
217     ushort leds;
218     if(kb->fwversion >= 0x300) {
219         leds = (kb->ileds << 8) | 0x0001;
220         ileds = &leds;
221     }
222     else {
223         ileds = &kb->ileds;
224     }
225     struct usbdevfs_ctrltransfer transfer = { 0x21, 0x09, 0x0200, 0x00, (kb->
      fwversion >= 0x300 ? 2 : 1), 500, ileds };
226     int res = ioctl(kb->handle - 1, USBDEVFS_CONTROL, &transfer);
227     if(res <= 0) {
228         ckb_err("%s\n", res ? strerror(errno) : "No data written");
229         if (usb_tryreset(kb) == 0 && countForReset++ < 3) {
230             os_sendindicators(kb);
231         }
232     }
233 }
```

Here is the call graph for this function:



Here is the caller graph for this function:

9.43.3.6 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 about connecting interfaces
- Claim the USB interfaces

Todo in these modules a pullrequest is outstanding

< Try to reset the device and recall the function

< Don't do this endless in recursion

< os_setupusb() has a return value (used as boolean)

Definition at line 548 of file usb_linux.c.

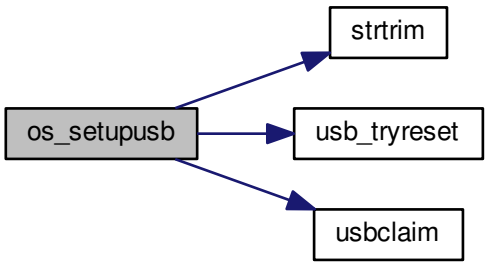
References ckb_err, ckb_info, devpath, usbdevice::epcount, usbdevice::fwversion, INDEX_OF, KB_NAME_LEN, keyboard, usbdevice::name, usbdevice::serial, SERIAL_LEN, strtrim(), usbdevice::udev, usb_tryreset(), and usbclaim().

Referenced by _setupusb().

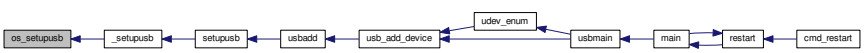
```
548                                {
551     struct udev_device* dev = kb->udev;
552     const char* name = udev_device_get_sysattr_value(dev, "product");
553     if(name)
```

```
554         strncpy(kb->name, name, KB_NAME_LEN);
555     strtrim(kb->name);
556     const char* serial = udev_device_get_sysattr_value(dev, "serial");
557     if(serial)
558         strncpy(kb->serial, serial, SERIAL_LEN);
559     strtrim(kb->serial);
560     const char* firmware = udev_device_get_sysattr_value(dev, "bcdDevice");
561     if(firmware)
562         sscanf(firmware, "%hx", &kb->fwversion);
563     else
564         kb->fwversion = 0;
565     int index = INDEX_OF(kb, keyboard);
566
567     ckb_info("Connecting %s at %s%d\n", kb->name, devpath, index);
568
569     const char* ep_str = udev_device_get_sysattr_value(dev, "bNumInterfaces");
570 #ifdef DEBUG
571     ckb_info("claiming interfaces. name=%s, firmware=%s; ep_str=%s\n", name, firmware, ep_str);
572 #endif //DEBUG
573     kb->epcount = 0;
574     if(ep_str)
575         sscanf(ep_str, "%d", &kb->epcount);
576     if(kb->epcount < 2){
577         // IF we have an RGB KB with 0 or 1 endpoints, it will be in BIOS mode.
578         ckb_err("Unable to read endpoint count from udev, assuming %d and reading >>%s<< or device
    is in BIOS mode\n", kb->epcount, ep_str);
579         if (usb_tryreset(kb) == 0) {
580             static int retryCount = 0;
581             if (retryCount++ < 5) {
582                 return os_setupusb(kb);
583             }
584         }
585         return -1;
586         // ToDo are there special versions we have to detect? If there are, that was the old code to handle
    it:
587         // This shouldn't happen, but if it does, assume EP count based onckb_warn what the device is
    supposed to have
588         // kb->epcount = (HAS_FEATURES(kb, FEAT_RGB) ? 4 : 3);
589         // ckb_warn("Unable to read endpoint count from udev, assuming %d and reading >>%s<<...\n",
    kb->epcount, ep_str);
590     }
591     if(usbclaim(kb)){
592         ckb_err("Failed to claim interfaces: %s\n", strerror(errno));
593         return -1;
594     }
595     return 0;
596 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



9.43.3.7 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 130 of file usb_linux.c.

References ckb_err_fn, ckb_warn_fn, usbdevice::epcount, usbdevice::handle, and MSG_SIZE.

Referenced by _usbrecv().

```
130                                                                         {
131     int res;
132     // This is what CUE does, but it doesn't seem to work on linux.
133     /*if(kb->fwversion >= 0x130){
134         struct usbdevfs_bulktransfer transfer = {0};
135         transfer.ep = 0x84;
136         transfer.len = MSG_SIZE;
137         transfer.timeout = 5000;
138         transfer.data = in_msg;
139         res = ioctl(kb->handle - 1, USBDEVFS_BULK, &transfer);
140     } else {*/
141         struct usbdevfs_ctrltransfer transfer = { 0xa1, 0x01, 0x0300, kb->
    epcount - 1, MSG_SIZE, 5000, in_msg };
142         res = ioctl(kb->handle - 1, USBDEVFS_CONTROL, &transfer);
143     //}
144     if(res <= 0){
145         ckb_err_fn("%s\n", file, line, res ? strerror(errno) : "No data read");
146         if(res == -1 && errno == ETIMEDOUT)
147             return -1;
148         else
149             return 0;
150     } else if(res != MSG_SIZE)
151         ckb_warn_fn("Read %d bytes (expected %d)\n", file, line, res,
    MSG_SIZE);
152 #ifdef DEBUG_USB_RECV
153     char converted[MSG_SIZE*3 + 1];
154     for(int i=0;i<MSG_SIZE;i++)
155         sprintf(&converted[i*3], "%02x ", in_msg[i]);
156     ckb_warn_fn("Recv %s\n", file, line, converted);
```

```
157 #endif
158     return res;
159 }
```

Here is the caller graph for this function:

9.43.3.8 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, Recipient=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_SEND is set during compilation, the number of bytes sent and their representation are logged to the error channel.

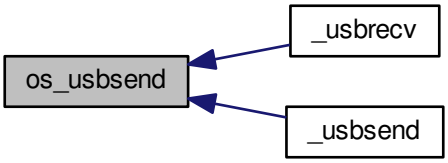
Definition at line 68 of file usb_linux.c.

References ckb_err_fn, ckb_warn_fn, usbdevice::epcount, usbdevice::fwversion, usbdevice::handle, IS_V2_OVERRIDE, and MSG_SIZE.

Referenced by _usbrecv(), and _usbsend().

```
68                                                                                                  {
69     int res;
70     if ((kb->fwversion >= 0x120 || IS_V2_OVERRIDE(kb)) && !is_recv){
71         struct usbdevfs_bulktransfer transfer = {0};
72         // FW 2.XX uses 0x03, FW 3.XX uses 0x02
73         transfer.ep = (kb->fwversion >= 0x130 && kb->fwversion < 0x200) ? 4 : (kb->
    fwversion >= 0x300 ? 2 : 3);
74         transfer.len = MSG_SIZE;
75         transfer.timeout = 5000;
76         transfer.data = (void*)out_msg;
77         res = ioctl(kb->handle - 1, USBDEVFS_BULK, &transfer);
78     } else {
79         struct usbdevfs_ctrltransfer transfer = { 0x21, 0x09, 0x0200, kb->
    epcount - 1, MSG_SIZE, 5000, (void*)out_msg };
80         res = ioctl(kb->handle - 1, USBDEVFS_CONTROL, &transfer);
81     }
82
83     if (res <= 0){
84         ckb_err_fn(" %s, res = 0x%x\n", file, line, res ? strerror(errno) : "No data written",
    res);
85         if(res == -1 && errno == ETIMEDOUT)
86             return -1;
87         else
88             return 0;
89     } else if (res != MSG_SIZE)
90         ckb_warn_fn("Wrote %d bytes (expected %d)\n", file, line, res,
    MSG_SIZE);
91 #ifdef DEBUG_USB_SEND
92     char converted[MSG_SIZE*3 + 1];
93     for(int i=0;i<MSG_SIZE;i++)
94         sprintf(&converted[i*3], "%02x ", out_msg[i]);
95     ckb_warn_fn("Sent %s\n", file, line, converted);
96 #endif
97     return res;
98 }
```

Here is the caller graph for this function:



9.43.3.9 void strtrim (char * *string*)

strtrim trims a string by removing leading and trailing spaces.

Parameters

string	

Definition at line 523 of file usb_linux.c.

Referenced by os_setupusb().

```
523                          {
524     // Find last non-space
525     char* last = string;
526     for(char* c = string; *c != 0; c++){
527         if(!isspace(*c))
528             last = c;
529     }
530     last[1] = 0;
531     // Find first non-space
532     char* first = string;
533     for(; *first != 0; first++){
534         if(!isspace(*first))
535             break;
536     }
537     if(first != string)
538         memmove(string, first, last - first);
539 }
```

Here is the caller graph for this function:

9.43.3.10 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 ().

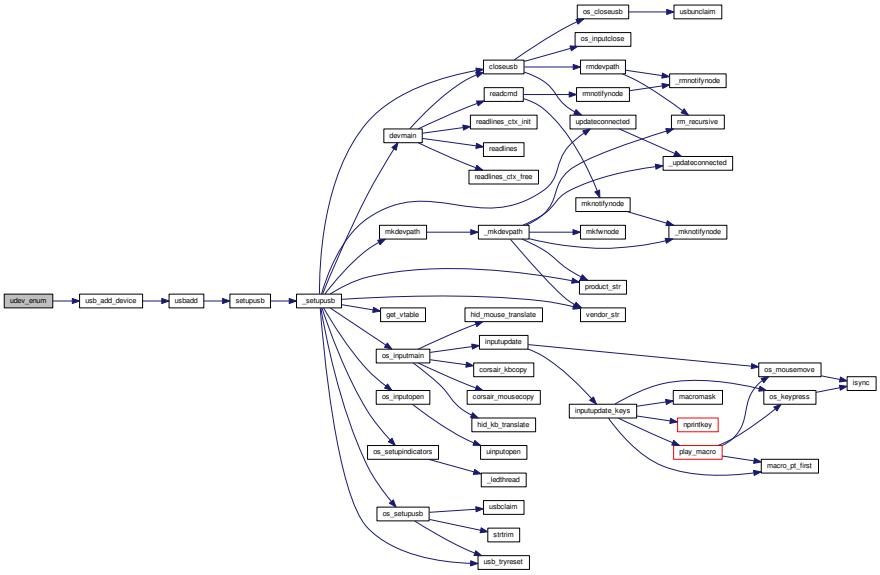
Definition at line 765 of file usb_linux.c.

References usb_add_device(), and V_CORSAIR_STR.

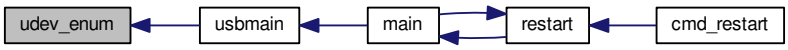
Referenced by usbmain().

```
765                         {
766     struct udev_enumerate* enumerator = udev_enumerate_new(udev);
767     udev_enumerate_add_match_subsystem(enumerator, "usb");
768     udev_enumerate_add_match_sysattr(enumerator, "idVendor", V_CORSAIR_STR);
769     udev_enumerate_scan_devices(enumerator);
770     struct udev_list_entry* devices, *dev_list_entry;
771     devices = udev_enumerate_get_list_entry(enumerator);
772
773     udev_list_entry_foreach(dev_list_entry, devices){
774         const char* path = udev_list_entry_get_name(dev_list_entry);
775         if(!path)
776             continue;
777         struct udev_device* dev = udev_device_new_from_syspath(udev, path);
778         if(!dev)
779             continue;
780         // If the device matches a recognized device ID, open it
781         if(usb_add_device(dev))
782             // Release device if not
783             udev_device_unref(dev);
784     }
785     udev_enumerate_unref(enumerator);
786 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



9.43.3.11 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.

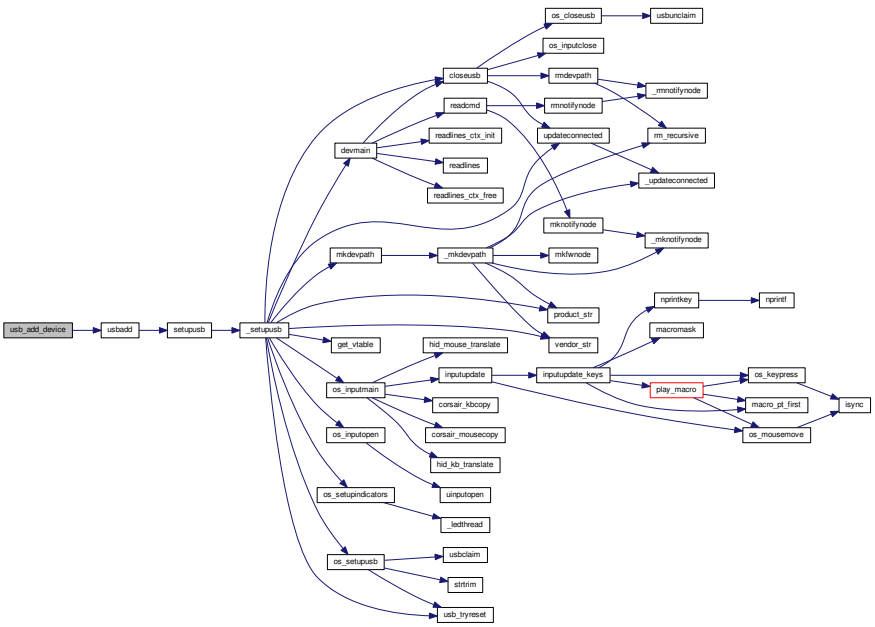
Definition at line 713 of file usb_linux.c.

References N_MODELS, usbadd(), V_CORSAIR, and V_CORSAIR_STR.

Referenced by udev_enum(), and usbmain().

```
713                                                 {
714     const char* vendor = udev_device_get_sysattr_value(dev, "idVendor");
715     if(vendor && !strcmp(vendor, V_CORSAIR_STR)){
716         const char* product = udev_device_get_sysattr_value(dev, "idProduct");
717         if(product){
718             for(_model* model = models; model < models +
    N_MODELS; model++){
719                 if(!strcmp(product, model->name)){
720                     return usbadd(dev, V_CORSAIR, model->number);
721                 }
722             }
723         }
724     }
725     return 1;
726 }
```

Here is the call graph for this function:



Here is the caller graph for this function:

9.43.3.12 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.

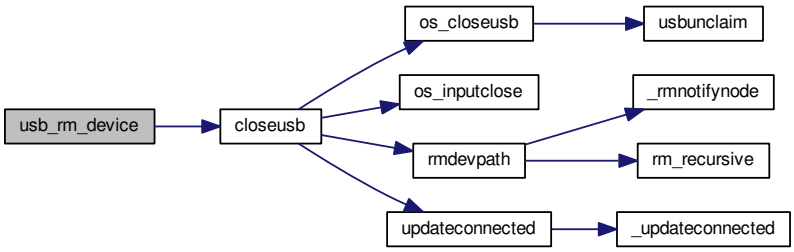
Definition at line 738 of file usb_linux.c.

References closeusb(), DEV_MAX, devmutex, kbsyspath, and keyboard.

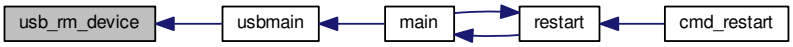
Referenced by usbmain().

```
738                                                 {
739     // Device removed. Look for it in our list of keyboards
740     const char* syspath = udev_device_get_syspath(dev);
741     if(!syspath || syspath[0] == 0)
742         return;
743     for(int i = 1; i < DEV_MAX; i++){
744         pthread_mutex_lock(devmutex + i);
745         if(!strcmp(syspath, kbsyspath[i]))
746             closeusb(keyboard + i);
747         pthread_mutex_unlock(devmutex + i);
748     }
749 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



9.43.3.13 int usbadd (struct udev_device * *dev*, short *vendor*, short *product*)

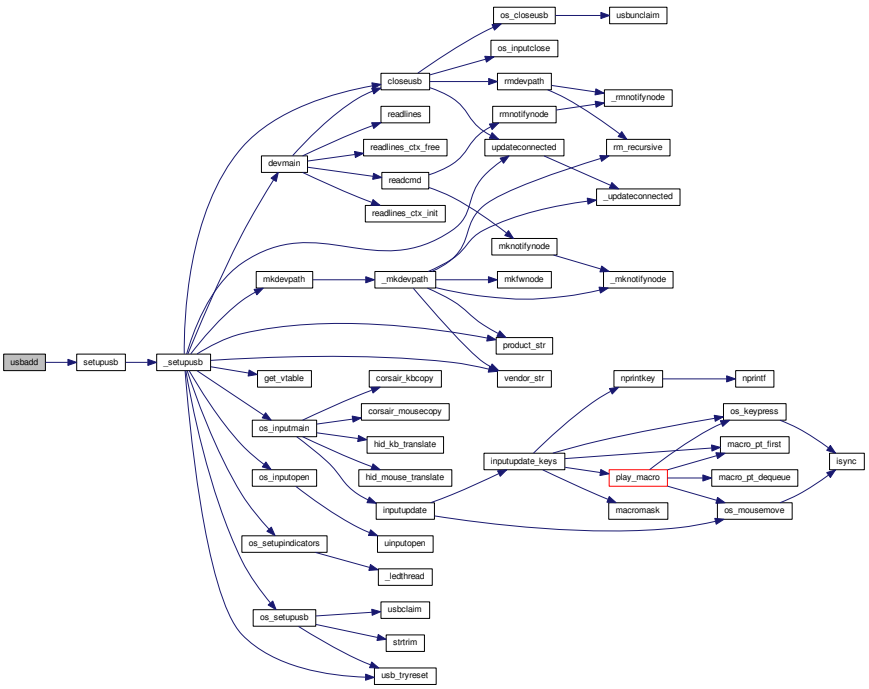
Definition at line 606 of file usb_linux.c.

References ckb_err, ckb_info, DEV_MAX, dmutex, usbdevice::handle, IS_CONNECTED, kbsyspath, keyboard, usbdevice::product, setupusb(), usbdevice::udev, and usbdevice::vendor.

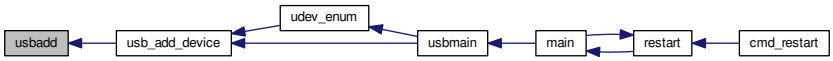
Referenced by usb_add_device().

```
606                                                                 {
607     const char* path = udev_device_get_devnode(dev);
608     const char* syspath = udev_device_get_syspath(dev);
609     if(!path || !syspath || path[0] == 0 || syspath[0] == 0){
610         ckb_err("Failed to get device path\n");
611         return -1;
612     }
613 #ifdef DEBUG
614     ckb_info(">>>vendor = 0x%x, product = 0x%x, path = %s, syspath = %s\n", vendor, product, path,
    syspath);
615 #endif // DEDBUG
616     // Find a free USB slot
617     for(int index = 1; index < DEV_MAX; index++){
618         usbdevice* kb = keyboard + index;
619         if(pthread_mutex_trylock(dmutex(kb))){
620             // If the mutex is locked then the device is obviously in use, so keep going
621             if(!strcmp(syspath, kbsyspath[index])){
622                 // Make sure this existing keyboard doesn't have the same syspath (this shouldn't happen)
623                 return 0;
624             }
625             continue;
626         }
627         if(!IS_CONNECTED(kb)){
628             // Open the sysfs device
629             kb->handle = open(path, O_RDWR) + 1;
630             if(kb->handle <= 0){
631                 ckb_err("Failed to open USB device: %s\n", strerror(errno));
632                 kb->handle = 0;
633                 pthread_mutex_unlock(dmutex(kb));
634                 return -1;
635             } else {
636                 // Set up device
637                 kb->udev = dev;
638                 kb->vendor = vendor;
639                 kb->product = product;
640                 strncpy(kbsyspath[index], syspath, FILENAME_MAX);
641                 // Mutex remains locked
642                 setupusb(kb);
643                 return 0;
644             }
645         }
646         pthread_mutex_unlock(dmutex(kb));
647     }
648     ckb_err("No free devices\n");
649     return -1;
650 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



9.43.3.14 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_DISCONNECT) 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 472 of file usb_linux.c.

References ckb_err, ckb_info, usbdevice::epcount, and usbdevice::handle.

Referenced by os_resetusb(), and os_setupusb().

```
472                                 {
473     int count = kb->epcount;
474 #ifdef DEBUG
475     ckb_info("claiming %d endpoints\n", count);
476 #endif // DEBUG
477
478     for(int i = 0; i < count; i++){
479         struct usbdevfs_ioctl ctl = { i, USBDEVFS_DISCONNECT, 0 };
480         ioctl(kb->handle - 1, USBDEVFS_IOCTL, &ctl);
481         if(ioctl(kb->handle - 1, USBDEVFS_CLAIMINTERFACE, &i)) {
482             ckb_err("Failed to claim interface %d: %s\n", i, strerror(errno));
483             return -1;
484         }
485     }
486     return 0;
487 }
```

Here is the caller graph for this function:

9.43.3.15 void usbkill ( )

Definition at line 853 of file usb_linux.c.

Referenced by quitWithLock().

```
853                 {
854     udev_unref(udev);
855     udev = 0;
856 }
```

Here is the caller graph for this function:

9.43.3.16 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...

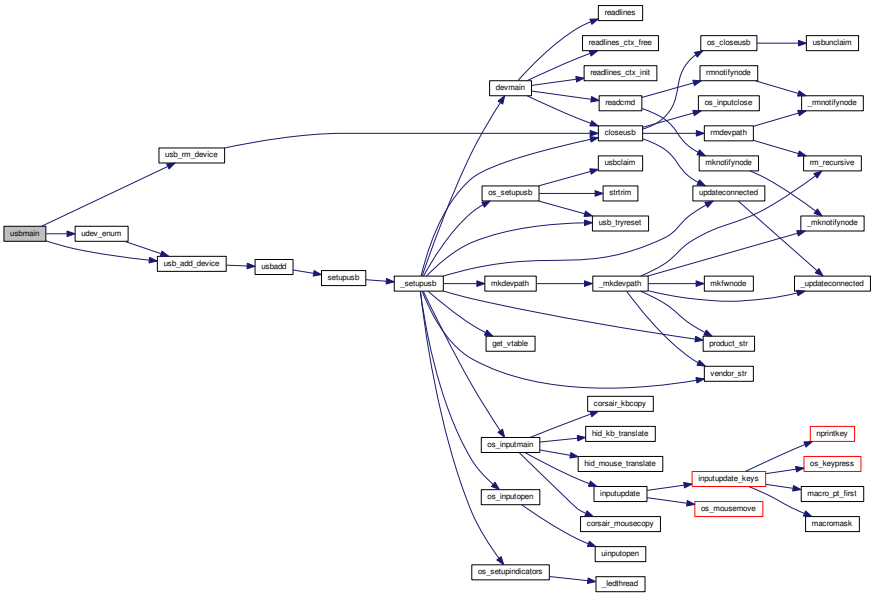
Definition at line 793 of file usb_linux.c.

References ckb_fatal, ckb_warn, udev_enum(), usb_add_device(), and usb_rm_device().

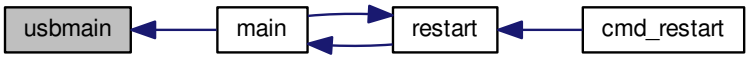
Referenced by main().

```
793             {
798     // Load the uinput module (if it's not loaded already)
799     if(system("modprobe uinput") != 0)
800         ckb_warn("Failed to load uinput module\n");
801
805     if(!(udev = udev_new())) {
806         ckb_fatal("Failed to initialize udev in usbmain(), usb_linux.c\n");
807         return -1;
808     }
809
812     udev_enum();
813
816     // Done scanning. Enter a loop to poll for device updates
817     struct udev_monitor* monitor = udev_monitor_new_from_netlink(udev, "udev");
818     udev_monitor_filter_add_match_subsystem_devtype(monitor, "usb", 0);
819     udev_monitor_enable_receiving(monitor);
820     // Get an fd for the monitor
821     int fd = udev_monitor_get_fd(monitor);
822     fd_set fds;
823     while(udev){
824         FD_ZERO(&fds);
825         FD_SET(fd, &fds);
826         // Block until an event is read
827         if(select(fd + 1, &fds, 0, 0, 0) > 0 && FD_ISSET(fd, &fds)){
828             struct udev_device* dev = udev_monitor_receive_device(monitor);
829             if(!dev)
830                 continue;
831             const char* action = udev_device_get_action(dev);
832             if(!action){
833                 udev_device_unref(dev);
834                 continue;
835             }
836             // Add/remove device
837             if(!strcmp(action, "add")){
838                 int res = usb_add_device(dev);
839                 if(res == 0)
840                     continue;
841                 // If the device matched but the handle wasn't opened correctly, re-enumerate (this
    sometimes solves the problem)
842                 if(res == -1)
843                     udev_enum();
844             } else if(!strcmp(action, "remove"))
845                 usb_rm_device(dev);
846             udev_device_unref(dev);
847         }
848     }
849     udev_monitor_unref(monitor);
850     return 0;
851 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



9.43.3.17 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.

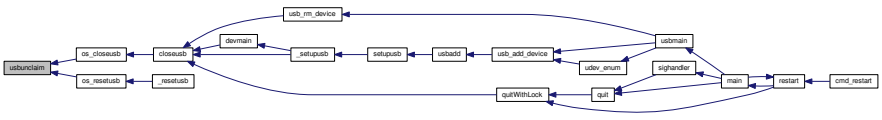
Definition at line 419 of file usb_linux.c.

References usbdevice::epcount, FEAT_RGB, usbdevice::handle, and HAS_FEATURES.

Referenced by os_closeusb(), and os_resetusb().

```
419                                                 {
420     int handle = kb->handle - 1;
421     int count = kb->epcount;
422     for (int i = 0; i < count; i++) {
423         ioctl(handle, USBDEVFS_RELEASEINTERFACE, &i);
424     }
425     // 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.
426     // Reconnecting any of the others causes trouble.
427     if (!resetting) {
428         struct usbdevfs_ioctl ctl = { 0, USBDEVFS_CONNECT, 0 };
429         ioctl(handle, USBDEVFS_IOCTL, &ctl);
430         ctl.ifno = 1;
431         ioctl(handle, USBDEVFS_IOCTL, &ctl);
432         // Also reconnect iface #2 (HID) for non-RGB keyboards
433         if(!HAS_FEATURES(kb, FEAT_RGB)){
434             ctl.ifno = 2;
435             ioctl(handle, USBDEVFS_IOCTL, &ctl);
436         }
437     }
438     return 0;
439 }
```

Here is the caller graph for this function:



9.43.4 Variable Documentation

9.43.4.1 char kbsyspath[9][FILENAME_MAX] [static]

Definition at line 13 of file usb_linux.c.

Referenced by os_closeusb(), usb_rm_device(), and usbadd().

9.43.4.2 _model models[] [static]

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 668 of file usb_linux.c.

9.43.4.3 struct udev* udev [static]

Definition at line 652 of file usb_linux.c.

9.43.4.4 pthread_t udevthread

Definition at line 655 of file usb_linux.c.

9.43.4.5 pthread_t usbthread

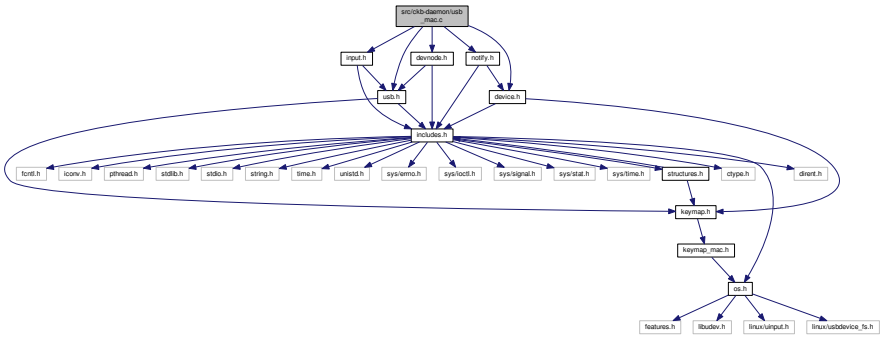
Todo These two thread vasriables seem to be unused: usbtread, udevthread

Definition at line 655 of file usb_linux.c.

9.44 src/ckb-daemon/usb_mac.c File Reference

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

Include dependency graph for usb_mac.c:



Index

_model, 356
_readlines_ctx, 71
_DEFAULT_SOURCE
    os.h, 238
_GNU_SOURCE
    os.h, 238
__FILE_NOPATH__
    includes.h, 112
_cmd_get
    notify.c, 226
_cmd_macro
    input.c, 116
_freeprofile
    profile.c, 240
_ledthread
    input_linux.c, 140
_mkdevpath
    devnode.c, 71
_mknotifynode
    devnode.c, 74
_nk95cmd
    usb.h, 327
    usb_linux.c, 357
_resetusb
    usb.c, 296
    usb.h, 329
_rmnotifynode
    devnode.c, 75
_setupusb
    usb.c, 296
_start_dev
    device.c, 46
_updateconnected
    devnode.c, 75
_usbrecv
    usb.c, 300
    usb.h, 329
_usbsend
    usb.c, 302
    usb.h, 331

ACCEL
    command.h, 39
ACTIVE
    command.h, 39
ACT_LIGHT
    device.h, 50
ACT_LOCK
    device.h, 50
ACT_M1
    device.h, 50
ACT_M2
    device.h, 51
ACT_M3
    device.h, 51
ACT_MR_RING
    device.h, 51
ACT_NEXT
    device.h, 51
ACT_NEXT_NOWRAP
    device.h, 51
active
    devcmd.__unnamed__, 28
allocprofile
    devcmd.__unnamed__, 28
    profile.c, 241
    profile.h, 257

BIND
    command.h, 40
BR1
    led_keyboard.c, 201
BR2
    led_keyboard.c, 201
BR4
    led_keyboard.c, 201
BTN_WHEELDOWN
    keymap.h, 154
BTN_WHEELUP
    keymap.h, 154
BUILD.md, 31
BUTTON_HID_COUNT
    keymap.c, 148
bind
    devcmd.__unnamed__, 28
binding, 281
bit_reverse_table
    led_keyboard.c, 210

CMD_FIRST
    command.h, 39
CMD_LAST
    command.h, 40
CMD_VT_FIRST
    command.h, 39
CLEAR_KEYBIT
    structures.h, 290
CMD_COUNT
    command.h, 38
CMD_DEV_COUNT

command.h, 38
ckb_err
includes.h, 112
ckb_err_fn
includes.h, 112
ckb_err_nofile
includes.h, 112
ckb_fatal
includes.h, 112
ckb_fatal_fn
includes.h, 112
ckb_fatal_nofile
includes.h, 112
ckb_info
includes.h, 112
ckb_info_fn
includes.h, 112
ckb_info_nofile
includes.h, 113
ckb_s_err
includes.h, 113
ckb_s_out
includes.h, 113
ckb_warn
includes.h, 113
ckb_warn_fn
includes.h, 113
ckb_warn_nofile
includes.h, 113
closeusb
usb.c, 303
usb.h, 333
cmd
command.h, 39
cmd_active_kb
device.h, 52
device_keyboard.c, 60
cmd_active_mouse
device.h, 53
device_mouse.c, 64
cmd_bind
input.c, 118
input.h, 131
cmd_dpi
dpi.c, 91
dpi.h, 97
cmd_dpisel
dpi.c, 92
dpi.h, 98
cmd_erase
profile.c, 241
profile.h, 258
cmd_eraseprofile
profile.c, 242
profile.h, 258
cmd_fwupdate
firmware.c, 104
firmware.h, 108
cmd_get
notify.c, 228
notify.h, 234
cmd_hwload_kb
profile.h, 259
profile_keyboard.c, 271
cmd_hwload_mouse
profile.h, 260
profile_mouse.c, 274
cmd_hwsave_kb
profile.h, 261
profile_keyboard.c, 272
cmd_hwsave_mouse
profile.h, 262
profile_mouse.c, 275
cmd_iauto
led.c, 161
led.h, 167
cmd_id
profile.c, 242
profile.h, 263
cmd_idle_kb
device.h, 53
device_keyboard.c, 60
cmd_idle_mouse
device.h, 54
device_mouse.c, 65
cmd_inotify
led.c, 162
led.h, 168
cmd_io_none
device_vtable.c, 68
cmd_ioff
led.c, 162
led.h, 168
cmd_ion
led.c, 163
led.h, 169
cmd_lift
dpi.c, 92
dpi.h, 98
cmd_macro
input.c, 118
input.h, 132
cmd_macro_none
device_vtable.c, 68
cmd_name
profile.c, 243
profile.h, 263
cmd_none
device_vtable.c, 68
cmd_notify
notify.c, 229
notify.h, 234
cmd_pollrate
device.h, 54
device_mouse.c, 65
cmd_profileid

profile.c, 243
profile.h, 264
cmd_profilename
profile.c, 244
profile.h, 264
cmd_rebind
input.c, 118
input.h, 132
cmd_restart
notify.c, 229
notify.h, 235
cmd_rgb
led.c, 163
led.h, 169
cmd_snap
dpi.c, 92
dpi.h, 98
cmd_strings
command.c, 36
cmd_unbind
input.c, 119
input.h, 133
cmdhandler
command.h, 39
cmdhandler_io
command.h, 39
cmdhandler_mac
command.h, 39
command.h
ACCEL, 39
ACTIVE, 39
BIND, 40
CMD_FIRST, 39
CMD_LAST, 40
CMD_VT_FIRST, 39
DELAY, 39
DITHER, 39
DPI, 40
DPISEL, 40
ERASE, 39
ERASEPROFILE, 39
FPS, 39
FWUPDATE, 39
GET, 40
HWLOAD, 39
HWSAVE, 39
IAUTO, 40
ID, 40
IDLE, 39
INOTIFY, 40
IOFF, 40
ION, 40
LAYOUT, 39
LIFT, 40
MACRO, 40
MODE, 39
NAME, 40
NONE, 39
NOTIFY, 40
NOTIFYOFF, 39
NOTIFYON, 39
POLLRATE, 39
PROFILEID, 40
PROFILENAME, 40
REBIND, 40
RESTART, 40
RGB, 40
SCROLLSPEED, 39
SNAP, 40
SWITCH, 39
UNBIND, 40
command.c
cmd_strings, 36
readcmd, 32
TRY_WITH_RESET, 32
command.h
CMD_COUNT, 38
CMD_DEV_COUNT, 38
cmd, 39
cmdhandler, 39
cmdhandler_io, 39
cmdhandler_mac, 39
devcmd, 39
readcmd, 41
corsair_kbcopy
keymap.c, 148
keymap.h, 156
corsair_mousecopy
keymap.c, 148
keymap.h, 157

DELAY
command.h, 39
DITHER
command.h, 39
DPI
command.h, 40
DPISEL
command.h, 40
DAEMON.md, 31
DEBUG
usb_linux.c, 356
DELAY_LONG
usb.h, 317
DELAY_MEDIUM
usb.h, 317
DELAY_SHORT
usb.h, 317
DEV_MAX
device.h, 51
DPI_COUNT
structures.h, 290
devcmd, 38
command.h, 39
devcmd.__unnamed__, 27
active, 28
allocprofile, 28

bind, 28
dpi, 28
dpisel, 28
erase, 28
eraseprofile, 28
freeprofile, 28
fwupdate, 28
get, 28
hwload, 28
hwsave, 28
iauto, 29
id, 29
idle, 29
inotify, 29
ioff, 29
ion, 29
lift, 29
loadprofile, 29
macro, 29
name, 29
notify, 29
pollrate, 29
profileid, 29
profilename, 29
rebind, 29
restart, 29
rgb, 29
setmodeindex, 29
snap, 29
start, 29
unbind, 29
updatedpi, 29
updateindicators, 29
updatergb, 29
device.c
_start_dev, 46
devlistmutex, 48
devmutex, 48
hwload_mode, 48
inputmutex, 48
keyboard, 48
macromutex, 48
macromutex2, 49
macrovar, 49
start_dev, 48
device.h
ACT_LIGHT, 50
ACT_LOCK, 50
ACT_M1, 50
ACT_M2, 51
ACT_M3, 51
ACT_MR_RING, 51
ACT_NEXT, 51
ACT_NEXT_NOWRAP, 51
cmd_active_kb, 52
cmd_active_mouse, 53
cmd_idle_kb, 53
cmd_idle_mouse, 54
cmd_pollrate, 54
DEV_MAX, 51
devmutex, 59
dmutex, 51
IN_CORSAIR, 51
IN_HID, 51
IS_CONNECTED, 52
imutex, 51
inputmutex, 59
keyboard, 59
macromutex, 59
macromutex2, 59
macrovar, 59
mmutex, 52
mmutex2, 52
mvar, 52
setactive, 52
setactive_kb, 54
setactive_mouse, 56
setmodeindex_nrgb, 57
start_dev, 58
start_kb_nrgb, 58
device_keyboard.c
cmd_active_kb, 60
cmd_idle_kb, 60
setactive_kb, 61
setmodeindex_nrgb, 63
start_kb_nrgb, 63
device_mouse.c
cmd_active_mouse, 64
cmd_idle_mouse, 65
cmd_pollrate, 65
setactive_mouse, 65
device_vtable.c
cmd_io_none, 68
cmd_macro_none, 68
cmd_none, 68
int1_int_none, 68
int1_void_none, 68
loadprofile_none, 69
vtable_keyboard, 69
vtable_keyboard_nonrgb, 69
vtable_mouse, 69
devlistmutex
device.c, 48
devmain
usb.c, 305
devmutex
device.c, 48
device.h, 59
devnode.c
_mkdevpath, 71
_mknotifynode, 74
_rmnotifynode, 75
_updateconnected, 75
devpath, 83
gid, 83
MAX_BUFFER, 71

mkdevpath, 76
mkfwnode, 77
mknotifynode, 78
readlines, 78
readlines_ctx_free, 79
readlines_ctx_init, 80
rm_recursive, 80
rmdevpath, 80
rmnotifynode, 81
S_GID_READ, 71
updateconnected, 82
devnode.h
devpath, 91
gid, 91
mkdevpath, 85
mkfwnode, 85
mknotifynode, 86
readlines, 87
readlines_ctx, 85
readlines_ctx_free, 88
readlines_ctx_init, 88
rmdevpath, 88
rmnotifynode, 89
S_CUSTOM, 84
S_CUSTOM_R, 84
S_READ, 84
S_READDIR, 84
S_READWRITE, 84
updateconnected, 90
devpath
devnode.c, 83
devnode.h, 91
dmutex
device.h, 51
dpi
devcmd.__unnamed__, 28
dpi.c
cmd_dpi, 91
cmd_dpisel, 92
cmd_lift, 92
cmd_snap, 92
loaddpi, 93
printdpi, 94
savedpi, 94
updatedpi, 95
dpi.h
cmd_dpi, 97
cmd_dpisel, 98
cmd_lift, 98
cmd_snap, 98
loaddpi, 99
printdpi, 100
savedpi, 100
updatedpi, 101
dpisel
devcmd.__unnamed__, 28
dpiset, 282

ERASE
command.h, 39
ERASEPROFILE
command.h, 39
erase
devcmd.__unnamed__, 28
eraseprofile
devcmd.__unnamed__, 28
euid_guard_start
os.h, 239
euid_guard_stop
os.h, 239

FPS
command.h, 39
FWUPDATE
command.h, 39
FEAT_ADJRATE
structures.h, 290
FEAT_ANSI
structures.h, 290
FEAT_BIND
structures.h, 290
FEAT_COMMON
structures.h, 290
FEAT_FWUPDATE
structures.h, 290
FEAT_FWVERSION
structures.h, 290
FEAT_HWLOAD
structures.h, 291
FEAT_ISO
structures.h, 291
FEAT_LMASK
structures.h, 291
FEAT_MONOCHROME
structures.h, 291
FEAT_MOUSEACCEL
structures.h, 291
FEAT_NOTIFY
structures.h, 291
FEAT_POLLRATE
structures.h, 291
FEAT_RGB
structures.h, 291
FEAT_STD_NRGB
structures.h, 291
FEAT_STD_RGB
structures.h, 292
FW_MAXSIZE
firmware.c, 104
FW_NOFILE
firmware.c, 104
FW_OK
firmware.c, 104
FW_USBFAIL
firmware.c, 104
FW_WRONGDEV
firmware.c, 104
features_mask

main.c, 224
usb.c, 312
firmware.c
cmd_fwupdate, 104
FW_MAXSIZE, 104
FW_NOFILE, 104
FW_OK, 104
FW_USBFAIL, 104
FW_WRONGDEV, 104
fwupdate, 105
getfwversion, 107
firmware.h
cmd_fwupdate, 108
getfwversion, 109
freebind
input.c, 119
input.h, 133
freemode
profile.c, 244
freeprofile
devcmd.__unnamed__, 28
profile.c, 245
profile.h, 265
fwupdate
devcmd.__unnamed__, 28
firmware.c, 105

GET
command.h, 40
get
devcmd.__unnamed__, 28
get_vtable
usb.c, 307
getfwversion
firmware.c, 107
firmware.h, 109
gethwmodename
profile.c, 246
profile.h, 265
gethwprofilename
profile.c, 246
profile.h, 266
getid
profile.c, 247
profile.h, 267
getmodename
profile.c, 248
profile.h, 267
getprofilename
profile.c, 248
profile.h, 268
gid
devnode.c, 83
devnode.h, 91

HWLOAD
command.h, 39
HWSAVE
command.h, 39
HAS_ANY_FEATURE
structures.h, 292
HAS_FEATURES
structures.h, 292
HW_STANDARD
notify.c, 225
HWMODE_K70
structures.h, 292
HWMODE_K95
structures.h, 292
HWMODE_MAX
structures.h, 292
HWMODE_OR_RETURN
notify.c, 225
has_key
led.c, 164
hid_kb_translate
keymap.c, 148
keymap.h, 157
hid_mouse_translate
keymap.c, 150
keymap.h, 159
hwload
devcmd.__unnamed__, 28
hwload_mode
device.c, 48
main.c, 224
usb.c, 312
hwloadmode
profile_keyboard.c, 273
hwloadprofile
profile.h, 257
hwprofile, 286
hwsave
devcmd.__unnamed__, 28
hwtonative
profile.c, 249
profile.h, 269

IAUTO
command.h, 40
ID
command.h, 40
IDLE
command.h, 39
INOTIFY
command.h, 40
IOFF
command.h, 40
ION
command.h, 40
I_CAPS
structures.h, 292
I_NUM
structures.h, 292
I_SCROLL
structures.h, 292
IFACE_MAX
structures.h, 292

IN_CORSAIR
device.h, 51
IN_HID
device.h, 51
INDEX_OF
includes.h, 113
IS_CONNECTED
device.h, 52
IS_FULLRANGE
usb.h, 318
IS_GLAIVE
usb.h, 318
IS_HARPOON
usb.h, 318
IS_K63
usb.h, 318
IS_K65
usb.h, 318
IS_K68
usb.h, 318
IS_K70
usb.h, 318
IS_K95
usb.h, 318
IS_M65
usb.h, 318
IS_MOD
input.h, 131
IS_MONOCHROME
usb.h, 319
IS_MONOCHROME_DEV
usb.h, 319
IS_MOUSE
usb.h, 319
IS_MOUSE_DEV
usb.h, 319
IS_PLATINUM
usb.h, 319
IS_RGB
usb.h, 319
IS_RGB_DEV
usb.h, 319
IS_SABRE
usb.h, 319
IS_SCIMITAR
usb.h, 319
IS_STRAFE
usb.h, 320
IS_V2_OVERRIDE
usb.h, 320
IS_WHEEL
input.c, 116
iauto
devcmd.__unnamed__, 29
id
devcmd.__unnamed__, 29
idle
devcmd.__unnamed__, 29
imutex
device.h, 51
includes.h
__FILE_NOPATH__, 112
ckb_err, 112
ckb_err_fn, 112
ckb_err_nofile, 112
ckb_fatal, 112
ckb_fatal_fn, 112
ckb_fatal_nofile, 112
ckb_info, 112
ckb_info_fn, 112
ckb_info_nofile, 113
ckb_s_err, 113
ckb_s_out, 113
ckb_warn, 113
ckb_warn_fn, 113
ckb_warn_nofile, 113
INDEX_OF, 113
timespec_add, 114
timespec_eq, 113
timespec_ge, 113
timespec_gt, 114
timespec_le, 114
timespec_lt, 114
uchar, 114
ushort, 114
initbind
input.c, 119
input.h, 133
initmode
profile.c, 250
inotify
devcmd.__unnamed__, 29
input.c
_cmd_macro, 116
cmd_bind, 118
cmd_macro, 118
cmd_rebind, 118
cmd_unbind, 119
freebind, 119
IS_WHEEL, 116
initbind, 119
inputupdate, 120
inputupdate_keys, 121
macro_pt_dequeue, 123
macro_pt_enqueue, 124
macro_pt_first, 124
macromask, 125
play_macro, 125
pt_head, 128
pt_tail, 128
updateindicators_kb, 127
input.h
cmd_bind, 131
cmd_macro, 132
cmd_rebind, 132
cmd_unbind, 133

freebind, 133
IS_MOD, 131
initbind, 133
inputupdate, 134
os_inputclose, 135
os_inputopen, 136
os_keypress, 137
os_mousemove, 137
os_setupindicators, 138
parameter_t, 131
ptlist_t, 131
updateindicators_kb, 139
input_linux.c
_ledthread, 140
isync, 141
os_inputclose, 141
os_inputopen, 142
os_keypress, 143
os_mousemove, 144
os_setupindicators, 145
uinputopen, 145
inputmutex
device.c, 48
device.h, 59
inputupdate
input.c, 120
input.h, 134
inputupdate_keys
input.c, 121
int1_int_none
device_vtable.c, 68
int1_void_none
device_vtable.c, 68
ioff
devcmd.__unnamed__, 29
ion
devcmd.__unnamed__, 29
isblack
led_mouse.c, 212
iselect
led.c, 164
isync
input_linux.c, 141

KB_NAME_LEN
structures.h, 293
KEY_BACKSLASH_ISO
keymap.h, 154
KEY_CORSAIR
keymap.h, 154
KEY_NONE
keymap.h, 154
KEY_UNBOUND
keymap.h, 154
kbsyspath
usb_linux.c, 379
key, 153
keyboard
device.c, 48
device.h, 59
keymacro, 280
keymap
keymap.c, 151
keymap.h, 160
keymap.c
BUTTON_HID_COUNT, 148
corsair_kbcopy, 148
corsair_mousecopy, 148
hid_kb_translate, 148
hid_mouse_translate, 150
keymap, 151
keymap.h
BTN_WHEELDOWN, 154
BTN_WHEELUP, 154
corsair_kbcopy, 156
corsair_mousecopy, 157
hid_kb_translate, 157
hid_mouse_translate, 159
KEY_BACKSLASH_ISO, 154
KEY_CORSAIR, 154
KEY_NONE, 154
KEY_UNBOUND, 154
keymap, 160
LED_DPI, 154
LED_MOUSE, 154
MOUSE_BUTTON_FIRST, 154
MOUSE_EXTRA_FIRST, 155
N_BUTTONS_EXTENDED, 155
N_BUTTONS_HW, 155
N_KEY_ZONES, 155
N_KEYBYTES_EXTENDED, 155
N_KEYBYTES_HW, 155
N_KEYBYTES_INPUT, 155
N_KEYS_EXTENDED, 155
N_KEYS_EXTRA, 155
N_KEYS_HW, 155
N_KEYS_INPUT, 156
N_MOUSE_ZONES, 156
SCAN_KBD, 156
SCAN_MOUSE, 156
SCAN_SILENT, 156

LAYOUT
command.h, 39
LIFT
command.h, 40
LED_DPI
keymap.h, 154
LED_MOUSE
keymap.h, 154
LIFT_MAX
structures.h, 293
LIFT_MIN
structures.h, 293
led.c
cmd_iauto, 161
cmd_inotify, 162
cmd_ioff, 162

cmd_ion, 163
cmd_rgb, 163
has_key, 164
iselect, 164
printrgb, 165
led.h
cmd_iauto, 167
cmd_inotify, 168
cmd_ioff, 168
cmd_ion, 169
cmd_rgb, 169
loadrgb_kb, 170
loadrgb_mouse, 172
printrgb, 172
savergb_kb, 174
savergb_mouse, 175
updatergb_kb, 176
updatergb_mouse, 177
led_keyboard.c
BR1, 201
BR2, 201
BR4, 201
bit_reverse_table, 210
loadrgb_kb, 202
makergb_512, 204
makergb_full, 205
O0, 201
O1, 201
O2, 201
O3, 202
O4, 202
O5, 202
O6, 202
O7, 202
O8, 202
ordered8to3, 205
quantize8to3, 206
rgbcmp, 206
savergb_kb, 207
updatergb_kb, 208
led_mouse.c
isblack, 212
loadrgb_mouse, 213
rgbcmp, 213
savergb_mouse, 214
updatergb_mouse, 214
lift
devcmd.__unnamed__, 29
lighting, 283
loaddpi
dpi.c, 93
dpi.h, 99
loadprofile
devcmd.__unnamed__, 29
profile.c, 250
profile.h, 269
loadprofile_none
device_vtable.c, 69
loadrgb_kb
led.h, 170
led_keyboard.c, 202
loadrgb_mouse
led.h, 172
led_mouse.c, 213
localecase
main.c, 216

MACRO
command.h, 40
MODE
command.h, 39
MACRO_MAX
structures.h, 293
MAX_BUFFER
devnode.c, 71
MD_NAME_LEN
structures.h, 293
MODE_COUNT
structures.h, 293
MOUSE_BUTTON_FIRST
keymap.h, 154
MOUSE_EXTRA_FIRST
keymap.h, 155
MSG_SIZE
structures.h, 293
macro
devcmd.__unnamed__, 29
macro_pt_dequeue
input.c, 123
macro_pt_enqueue
input.c, 124
macro_pt_first
input.c, 124
macroaction, 279
macromask
input.c, 125
macromutex
device.c, 48
device.h, 59
macromutex2
device.c, 49
device.h, 59
macrovar
device.c, 49
device.h, 59
main
main.c, 217
main.c
features_mask, 224
hwload_mode, 224
localecase, 216
main, 217
main_ac, 224
main_av, 224
quit, 219
quitWithLock, 220
reset_stop, 224

restart, 221
sighandler, 222
sighandler2, 223
timespec_add, 223
main_ac
main.c, 224
main_av
main.c, 224
makergb_512
led_keyboard.c, 204
makergb_full
led_keyboard.c, 205
mkdevpath
devnode.c, 76
devnode.h, 85
mkfwnode
devnode.c, 77
devnode.h, 85
mknotifynode
devnode.c, 78
devnode.h, 86
mmutex
device.h, 52
mmutex2
device.h, 52
models
usb_linux.c, 379
mvar
device.h, 52

NAME
command.h, 40
NONE
command.h, 39
NOTIFY
command.h, 40
NOTIFYOFF
command.h, 39
NOTIFYON
command.h, 39
N_BUTTONS_EXTENDED
keymap.h, 155
N_BUTTONS_HW
keymap.h, 155
N_KEY_ZONES
keymap.h, 155
N_KEYBYTES_EXTENDED
keymap.h, 155
N_KEYBYTES_HW
keymap.h, 155
N_KEYBYTES_INPUT
keymap.h, 155
N_KEYS_EXTENDED
keymap.h, 155
N_KEYS_EXTRA
keymap.h, 155
N_KEYS_HW
keymap.h, 155
N_KEYS_INPUT
keymap.h, 156
N_MODELS
usb_linux.c, 356
N_MOUSE_ZONES
keymap.h, 156
NEEDS_FW_UPDATE
structures.h, 293
NK95_HWOFF
usb.h, 320
NK95_HWON
usb.h, 320
NK95_M1
usb.h, 320
NK95_M2
usb.h, 320
NK95_M3
usb.h, 320
name
devcmd.__unnamed__, 29
nativetohw
profile.c, 251
profile.h, 270
nk95cmd
usb.h, 320
notify
devcmd.__unnamed__, 29
notify.c
_cmd_get, 226
cmd_get, 228
cmd_notify, 229
cmd_restart, 229
HW_STANDARD, 225
HWMODE_OR_RETURN, 225
nprintf, 230
nprintind, 230
nprintkey, 231
restart, 232
notify.h
cmd_get, 234
cmd_notify, 234
cmd_restart, 235
nprintf, 235
nprintind, 236
nprintkey, 237
nprintf
notify.c, 230
notify.h, 235
nprintind
notify.c, 230
notify.h, 236
nprintkey
notify.c, 231
notify.h, 237

O0
led_keyboard.c, 201
O1
led_keyboard.c, 201
O2

led_keyboard.c, 201
O3
led_keyboard.c, 202
O4
led_keyboard.c, 202
O5
led_keyboard.c, 202
O6
led_keyboard.c, 202
O7
led_keyboard.c, 202
O8
led_keyboard.c, 202
OUTFIFO_MAX
structures.h, 293
ordered8to3
led_keyboard.c, 205
os.h
_DEFAULT_SOURCE, 238
_GNU_SOURCE, 238
euid_guard_start, 239
euid_guard_stop, 239
UINPUT_VERSION, 239
os_closeusb
usb.h, 334
usb_linux.c, 358
os_inputclose
input.h, 135
input_linux.c, 141
os_inputmain
usb.h, 335
usb_linux.c, 358
os_inputopen
input.h, 136
input_linux.c, 142
os_keypress
input.h, 137
input_linux.c, 143
os_mousemove
input.h, 137
input_linux.c, 144
os_resetusb
usb.h, 339
usb_linux.c, 362
os_sendindicators
usb.h, 340
usb_linux.c, 363
os_setupindicators
input.h, 138
input_linux.c, 145
os_setupusb
usb.h, 341
usb_linux.c, 364
os_usbrecv
usb.h, 343
usb_linux.c, 365
os_usbsend
usb.h, 345
usb_linux.c, 367
POLLRATE
command.h, 39
PROFILEID
command.h, 40
PROFILENAME
command.h, 40
P_GLAIVE
usb.h, 321
P_GLAIVE_STR
usb.h, 321
P_HARPOON
usb.h, 321
P_HARPOON_STR
usb.h, 321
P_K63_NRGB
usb.h, 321
P_K63_NRGB_STR
usb.h, 321
P_K65
usb.h, 321
P_K65_LUX
usb.h, 321
P_K65_LUX_STR
usb.h, 321
P_K65_NRGB
usb.h, 321
P_K65_NRGB_STR
usb.h, 322
P_K65_RFIRE
usb.h, 322
P_K65_RFIRE_STR
usb.h, 322
P_K65_STR
usb.h, 322
P_K68
usb.h, 322
P_K68_STR
usb.h, 322
P_K70
usb.h, 322
P_K70_LUX
usb.h, 322
P_K70_LUX_NRGB
usb.h, 322
P_K70_LUX_NRGB_STR
usb.h, 322
P_K70_LUX_STR
usb.h, 323
P_K70_NRGB
usb.h, 323
P_K70_NRGB_STR
usb.h, 323
P_K70_RFIRE
usb.h, 323
P_K70_RFIRE_NRGB
usb.h, 323
P_K70_RFIRE_NRGB_STR

usb.h, 323
P_K70_RFIRE_STR
usb.h, 323
P_K70_STR
usb.h, 323
P_K95
usb.h, 323
P_K95_NRGB
usb.h, 323
P_K95_NRGB_STR
usb.h, 323
P_K95_PLATINUM
usb.h, 324
P_K95_PLATINUM_STR
usb.h, 324
P_K95_STR
usb.h, 324
P_M65
usb.h, 324
P_M65_PRO
usb.h, 324
P_M65_PRO_STR
usb.h, 324
P_M65_STR
usb.h, 324
P_SABRE_L
usb.h, 324
P_SABRE_L_STR
usb.h, 324
P_SABRE_N
usb.h, 324
P_SABRE_N_STR
usb.h, 324
P_SABRE_O
usb.h, 325
P_SABRE_O2
usb.h, 325
P_SABRE_O2_STR
usb.h, 325
P_SABRE_O_STR
usb.h, 325
P_SCIMITAR
usb.h, 325
P_SCIMITAR_PRO
usb.h, 325
P_SCIMITAR_PRO_STR
usb.h, 325
P_SCIMITAR_STR
usb.h, 325
P_STRAFE
usb.h, 325
P_STRAFE_NRGB
usb.h, 325
P_STRAFE_NRGB_2
usb.h, 326
P_STRAFE_NRGB_2_STR
usb.h, 326
P_STRAFE_NRGB_STR
usb.h, 326
P_STRAFE_STR
usb.h, 326
PR_NAME_LEN
structures.h, 294
parameter, 130
parameter_t
input.h, 131
play_macro
input.c, 125
pollrate
devcmd.__unnamed__, 29
printdpi
dpi.c, 94
dpi.h, 100
printname
profile.c, 251
printrgb
led.c, 165
led.h, 172
product_str
usb.c, 308
usb.h, 347
profile.c
_freeprofile, 240
allocprofile, 241
cmd_erase, 241
cmd_eraseprofile, 242
cmd_id, 242
cmd_name, 243
cmd_profileid, 243
cmd_profilename, 244
freemode, 244
freeprofile, 245
gethwmodename, 246
gethwprofilename, 246
getid, 247
getmodename, 248
getprofilename, 248
hwtonative, 249
initmode, 250
loadprofile, 250
nativetohw, 251
printname, 251
setid, 252
u16dec, 253
u16enc, 253
urldecode2, 254
urlencode2, 255
utf16to8, 255
utf8to16, 256
profile.h
allocprofile, 257
cmd_erase, 258
cmd_eraseprofile, 258
cmd_hwload_kb, 259
cmd_hwload_mouse, 260
cmd_hwsave_kb, 261

cmd_hwsave_mouse, 262
cmd_id, 263
cmd_name, 263
cmd_profileid, 264
cmd_profilename, 264
freeprofile, 265
gethwmodename, 265
gethwprofilename, 266
getid, 267
getmodename, 267
getprofilename, 268
hwloadprofile, 257
hwtonative, 269
loadprofile, 269
nativetohw, 270
setid, 270
profile_keyboard.c
cmd_hwload_kb, 271
cmd_hwsave_kb, 272
hwloadmode, 273
profile_mouse.c
cmd_hwload_mouse, 274
cmd_hwsave_mouse, 275
profileid
devcmd.__unnamed__, 29
profilename
devcmd.__unnamed__, 29
pt_head
input.c, 128
pt_tail
input.c, 128
ptlist, 131
ptlist_t
input.h, 131

quantize8to3
led_keyboard.c, 206
quit
main.c, 219
quitWithLock
main.c, 220

REBIND
command.h, 40
RESTART
command.h, 40
RGB
command.h, 40
README.md, 31
ROADMAP.md, 31
readcmd
command.c, 32
command.h, 41
readlines
devnode.c, 78
devnode.h, 87
readlines_ctx
devnode.h, 85
readlines_ctx_free
devnode.c, 79
devnode.h, 88
readlines_ctx_init
devnode.c, 80
devnode.h, 88
rebind
devcmd.__unnamed__, 29
reset_stop
main.c, 224
usb.c, 312
resetusb
usb.h, 326
restart
devcmd.__unnamed__, 29
main.c, 221
notify.c, 232
revertusb
usb.c, 309
usb.h, 348
rgb
devcmd.__unnamed__, 29
rgbcmp
led_keyboard.c, 206
led_mouse.c, 213
rm_recursive
devnode.c, 80
rmdevpath
devnode.c, 80
devnode.h, 88
rmnotifynode
devnode.c, 81
devnode.h, 89

SCROLLSPEED
command.h, 39
SNAP
command.h, 40
SWITCH
command.h, 39
S_CUSTOM
devnode.h, 84
S_CUSTOM_R
devnode.h, 84
S_GID_READ
devnode.c, 71
S_READ
devnode.h, 84
S_READDIR
devnode.h, 84
S_READWRITE
devnode.h, 84
SCAN_KBD
keymap.h, 156
SCAN_MOUSE
keymap.h, 156
SCAN_SILENT
keymap.h, 156
SCROLL_ACCELERATED
structures.h, 294

SCROLL_MAX
structures.h, 294
SCROLL_MIN
structures.h, 294
SERIAL_LEN
structures.h, 294
SET_KEYBIT
structures.h, 294
savedpi
dpi.c, 94
dpi.h, 100
savergb_kb
led.h, 174
led_keyboard.c, 207
savergb_mouse
led.h, 175
led_mouse.c, 214
setactive
device.h, 52
setactive_kb
device.h, 54
device_keyboard.c, 61
setactive_mouse
device.h, 56
device_mouse.c, 65
setid
profile.c, 252
profile.h, 270
setmodeindex
devcmd.__unnamed__, 29
setmodeindex_nrgb
device.h, 57
device_keyboard.c, 63
setupusb
usb.c, 310
usb.h, 349
sighandler
main.c, 222
sighandler2
main.c, 223
snap
devcmd.__unnamed__, 29
src/ckb-daemon/command.c, 31
src/ckb-daemon/command.h, 36
src/ckb-daemon/device.c, 45
src/ckb-daemon/device.h, 49
src/ckb-daemon/device_keyboard.c, 59
src/ckb-daemon/device_mouse.c, 64
src/ckb-daemon/device_vtable.c, 67
src/ckb-daemon/devnode.c, 69
src/ckb-daemon/devnode.h, 83
src/ckb-daemon/dpi.c, 91
src/ckb-daemon/dpi.h, 96
src/ckb-daemon/extra_mac.c, 102
src/ckb-daemon/firmware.c, 103
src/ckb-daemon/firmware.h, 108
src/ckb-daemon/includes.h, 110
src/ckb-daemon/input.c, 114
src/ckb-daemon/input.h, 128
src/ckb-daemon/input_linux.c, 140
src/ckb-daemon/input_mac.c, 146
src/ckb-daemon/input_mac_mouse.c, 147
src/ckb-daemon/keymap.c, 147
src/ckb-daemon/keymap.h, 151
src/ckb-daemon/keymap_mac.h, 160
src/ckb-daemon/led.c, 161
src/ckb-daemon/led.h, 166
src/ckb-daemon/led_keyboard.c, 178
src/ckb-daemon/led_mouse.c, 212
src/ckb-daemon/main.c, 216
src/ckb-daemon/notify.c, 224
src/ckb-daemon/notify.h, 233
src/ckb-daemon/os.h, 238
src/ckb-daemon/profile.c, 239
src/ckb-daemon/profile.h, 256
src/ckb-daemon/profile_keyboard.c, 271
src/ckb-daemon/profile_mouse.c, 274
src/ckb-daemon/structures.h, 276
src/ckb-daemon/usb.c, 295
src/ckb-daemon/usb.h, 313
src/ckb-daemon/usb_linux.c, 354
src/ckb-daemon/usb_mac.c, 380
start
devcmd.__unnamed__, 29
start_dev
device.c, 48
device.h, 58
start_kb_nrgb
device.h, 58
device_keyboard.c, 63
strtrim
usb_linux.c, 369
structures.h
CLEAR_KEYBIT, 290
DPI_COUNT, 290
FEAT_ADJRATE, 290
FEAT_ANSI, 290
FEAT_BIND, 290
FEAT_COMMON, 290
FEAT_FWUPDATE, 290
FEAT_FWVERSION, 290
FEAT_HWLOAD, 291
FEAT_ISO, 291
FEAT_LMASK, 291
FEAT_MONOCHROME, 291
FEAT_MOUSEACCEL, 291
FEAT_NOTIFY, 291
FEAT_POLLRATE, 291
FEAT_RGB, 291
FEAT_STD_NRGB, 291
FEAT_STD_RGB, 292
HAS_ANY_FEATURE, 292
HAS_FEATURES, 292
HWMODE_K70, 292
HWMODE_K95, 292
HWMODE_MAX, 292

I_CAPS, 292
I_NUM, 292
I_SCROLL, 292
IFACE_MAX, 292
KB_NAME_LEN, 293
LIFT_MAX, 293
LIFT_MIN, 293
MACRO_MAX, 293
MD_NAME_LEN, 293
MODE_COUNT, 293
MSG_SIZE, 293
NEEDS_FW_UPDATE, 293
OUTFIFO_MAX, 293
PR_NAME_LEN, 294
SCROLL_ACCELERATED, 294
SCROLL_MAX, 294
SCROLL_MIN, 294
SERIAL_LEN, 294
SET_KEYBIT, 294
vtable_keyboard, 294
vtable_keyboard_nonrgb, 294
vtable_mouse, 294

TEST_RESET
usb_linux.c, 356
TRY_WITH_RESET
command.c, 32
timespec_add
includes.h, 114
main.c, 223
timespec_eq
includes.h, 113
timespec_ge
includes.h, 113
timespec_gt
includes.h, 114
timespec_le
includes.h, 114
timespec_lt
includes.h, 114

u16dec
profile.c, 253
u16enc
profile.c, 253
UNBIND
command.h, 40
UINPUT_VERSION
os.h, 239
USB_DELAY_DEFAULT
usb.h, 326
uchar
includes.h, 114
udev
usb_linux.c, 379
udev_enum
usb_linux.c, 369
udevthread
usb_linux.c, 379
uinputopen
input_linux.c, 145
unbind
devcmd.__unnamed__, 29
updateconnected
devnode.c, 82
devnode.h, 90
updatedpi
devcmd.__unnamed__, 29
dpi.c, 95
dpi.h, 101
updateindicators
devcmd.__unnamed__, 29
updateindicators_kb
input.c, 127
input.h, 139
updatergb
devcmd.__unnamed__, 29
updatergb_kb
led.h, 176
led_keyboard.c, 208
updatergb_mouse
led.h, 177
led_mouse.c, 214
urldecode2
profile.c, 254
urlencode2
profile.c, 255
usb.c
_resetusb, 296
_setupusb, 296
_usbrecv, 300
_usbsend, 302
closeusb, 303
devmain, 305
features_mask, 312
get_vtable, 307
hwload_mode, 312
product_str, 308
reset_stop, 312
revertusb, 309
setupusb, 310
usb_tryreset, 311
usbmutex, 313
vendor_str, 312
usb.h
_nk95cmd, 327
_resetusb, 329
_usbrecv, 329
_usbsend, 331
closeusb, 333
DELAY_LONG, 317
DELAY_MEDIUM, 317
DELAY_SHORT, 317
IS_FULLRANGE, 318
IS_GLAIVE, 318
IS_HARPOON, 318
IS_K63, 318

IS_K65, 318
IS_K68, 318
IS_K70, 318
IS_K95, 318
IS_M65, 318
IS_MONOCHROME, 319
IS_MONOCHROME_DEV, 319
IS_MOUSE, 319
IS_MOUSE_DEV, 319
IS_PLATINUM, 319
IS_RGB, 319
IS_RGB_DEV, 319
IS_SABRE, 319
IS_SCIMITAR, 319
IS_STRAFE, 320
IS_V2_OVERRIDE, 320
NK95_HWOFF, 320
NK95_HWON, 320
NK95_M1, 320
NK95_M2, 320
NK95_M3, 320
nk95cmd, 320
os_closeusb, 334
os_inputmain, 335
os_resetusb, 339
os_sendindicators, 340
os_setupusb, 341
os_usbrecv, 343
os_usbsend, 345
P_GLAIVE, 321
P_GLAIVE_STR, 321
P_HARPOON, 321
P_HARPOON_STR, 321
P_K63_NRGB, 321
P_K63_NRGB_STR, 321
P_K65, 321
P_K65_LUX, 321
P_K65_LUX_STR, 321
P_K65_NRGB, 321
P_K65_NRGB_STR, 322
P_K65_RFIRE, 322
P_K65_RFIRE_STR, 322
P_K65_STR, 322
P_K68, 322
P_K68_STR, 322
P_K70, 322
P_K70_LUX, 322
P_K70_LUX_NRGB, 322
P_K70_LUX_NRGB_STR, 322
P_K70_LUX_STR, 323
P_K70_NRGB, 323
P_K70_NRGB_STR, 323
P_K70_RFIRE, 323
P_K70_RFIRE_NRGB, 323
P_K70_RFIRE_STR, 323
P_K70_STR, 323
P_K95, 323
P_K95_NRGB, 323
P_K95_NRGB_STR, 323
P_K95_PLATINUM, 324
P_K95_PLATINUM_STR, 324
P_K95_STR, 324
P_M65, 324
P_M65_PRO, 324
P_M65_PRO_STR, 324
P_M65_STR, 324
P_SABRE_L, 324
P_SABRE_L_STR, 324
P_SABRE_N, 324
P_SABRE_N_STR, 324
P_SABRE_O, 325
P_SABRE_O2, 325
P_SABRE_O2_STR, 325
P_SABRE_O_STR, 325
P_SCIMITAR, 325
P_SCIMITAR_PRO, 325
P_SCIMITAR_PRO_STR, 325
P_SCIMITAR_STR, 325
P_STRAFE, 325
P_STRAFE_NRGB, 325
P_STRAFE_NRGB_2, 326
P_STRAFE_NRGB_STR, 326
P_STRAFE_STR, 326
product_str, 347
resetusb, 326
revertusb, 348
setupusb, 349
USB_DELAY_DEFAULT, 326
usb_tryreset, 350
usbkill, 351
usbmain, 352
usbrecv, 326
usbsend, 326
V_CORSAIR, 327
V_CORSAIR_STR, 327
vendor_str, 353
usb_add_device
usb_linux.c, 370
usb_linux.c
_nk95cmd, 357
DEBUG, 356
kbsyspath, 379
models, 379
N_MODELS, 356
os_closeusb, 358
os_inputmain, 358
os_resetusb, 362
os_sendindicators, 363
os_setupusb, 364
os_usbrecv, 365
os_usbsend, 367
strtrim, 369
TEST_RESET, 356
udev, 379
udev_enum, 369
udevthread, 379

usb_add_device, 370
usb_rm_device, 373
usbadd, 374
usbclaim, 375
usbkill, 376
usbmain, 376
usbthread, 379
usbunclaim, 378
usb_rm_device
usb_linux.c, 373
usb_tryreset
usb.c, 311
usb.h, 350
usbadd
usb_linux.c, 374
usbclaim
usb_linux.c, 375
usbdevice, 288
usbid, 278
usbinput, 287
usbkill
usb.h, 351
usb_linux.c, 376
usbmain
usb.h, 352
usb_linux.c, 376
usbmode, 284
usbmutex
usb.c, 313
usbprofile, 285
usbrecv
usb.h, 326
usbsend
usb.h, 326
usbthread
usb_linux.c, 379
usbunclaim
usb_linux.c, 378
ushort
includes.h, 114
utf16to8
profile.c, 255
utf8to16
profile.c, 256

V_CORSAIR
usb.h, 327
V_CORSAIR_STR
usb.h, 327
vendor_str
usb.c, 312
usb.h, 353
vtable_keyboard
device_vtable.c, 69
structures.h, 294
vtable_keyboard_nonrgb
device_vtable.c, 69
structures.h, 294
vtable_mouse
device_vtable.c, 69
structures.h, 294