# ckb-next v0.2.8 at branch master

Generated by Doxygen 1.8.6

Wed May 24 2017 08:16:55

# **Contents**

1	ckb-next: RGB Driver for Linux and OS X				1	
2	Buil	Building ckb				
3	cbk	Improve	ements Ro	padmap	11	
4	DAE	MON			13	
5	Data	Struct	ure Index		21	
	5.1	Data S	tructures		. 21	
6	Eilo	Index			23	
0	6.1	File Lis	<b>,</b> +			
	0.1	FIIE LIS	il		. 23	
7	Data	Struct	ure Docun	nentation	25	
	7.1	devcm	dunnan	ned Struct Reference	. 25	
		7.1.1	Detailed	Description	. 26	
		7.1.2	Field Doo	cumentation	. 26	
			7.1.2.1	active	. 26	
			7.1.2.2	allocprofile	. 26	
			7.1.2.3	bind	. 26	
			7.1.2.4	dpi	. 26	
			7.1.2.5	dpisel	. 26	
			7.1.2.6	erase	. 26	
			7.1.2.7	eraseprofile	. 26	
			7.1.2.8	freeprofile	. 26	
			7.1.2.9	fwupdate	. 26	
			7.1.2.10	get	. 26	
			7.1.2.11	hwload	. 26	
			7.1.2.12	hwsave	. 27	
			7.1.2.13	iauto	. 27	
			7.1.2.14	$id \ldots \ldots$	. 27	
			7.1.2.15	idle	. 27	
			71216	inotify	27	

iv CONTENTS

			7.1.2.17 loft	27
			7.1.2.18 ion	27
			7.1.2.19 lift	27
			7.1.2.20 loadprofile	27
			7.1.2.21 macro	27
			7.1.2.22 name	27
			7.1.2.23 notify	27
			7.1.2.24 pollrate	27
			7.1.2.25 profileid	27
			7.1.2.26 profilename	27
			7.1.2.27 rebind	27
			7.1.2.28 restart	27
			7.1.2.29 rgb	27
			7.1.2.30 setmodeindex	27
			7.1.2.31 snap	27
			7.1.2.32 start	27
			7.1.2.33 unbind	27
			7.1.2.34 updatedpi	27
			7.1.2.35 updateindicators	27
			7.1.2.36 updatergb	27
8	File	Docum	entation	29
•				
	8.1	BUII D	md File Reference	29
	8.1 8.2		md File Reference	29 29
	8.2	DAEM	ON.md File Reference	29
	8.2 8.3	DAEM	ON.md File Reference	29 29
	8.2 8.3 8.4	DAEM READI ROAD	ON.md File Reference	29 29 29
	8.2 8.3	DAEMI READI ROAD src/ckb	ON.md File Reference	29 29 29 29
	8.2 8.3 8.4	DAEM READI ROAD	ON.md File Reference  ME.md File Reference  MAP.md File Reference  -daemon/command.c File Reference  Macro Definition Documentation	29 29 29 29 30
	8.2 8.3 8.4	DAEMOREADI ROADI src/ckb	ME.md File Reference  ME.md File Reference  MAP.md File Reference  -daemon/command.c File Reference  Macro Definition Documentation  8.5.1.1 TRY_WITH_RESET	29 29 29 29 30 30
	8.2 8.3 8.4	DAEMI READI ROAD src/ckb	DN.md File Reference  ME.md File Reference  MAP.md File Reference  -daemon/command.c File Reference  Macro Definition Documentation  8.5.1.1 TRY_WITH_RESET  Function Documentation	29 29 29 30 30 30
	8.2 8.3 8.4	DAEMOREADI ROADI src/ckb 8.5.1	ME.md File Reference  MAP.md File Reference  -daemon/command.c File Reference  Macro Definition Documentation  8.5.1.1 TRY_WITH_RESET  Function Documentation  8.5.2.1 readcmd	29 29 29 30 30 30 30
	8.2 8.3 8.4	DAEMOREADI ROADI src/ckb	ON.md File Reference  ME.md File Reference  MAP.md File Reference  -daemon/command.c File Reference  Macro Definition Documentation  8.5.1.1 TRY_WITH_RESET  Function Documentation  8.5.2.1 readcmd  Variable Documentation	29 29 29 30 30 30
	8.2 8.3 8.4	READI ROADI src/ckb 8.5.1 8.5.2 8.5.3	ON.md File Reference  ME.md File Reference  MAP.md File Reference  -daemon/command.c File Reference  Macro Definition Documentation  8.5.1.1 TRY_WITH_RESET  Function Documentation  8.5.2.1 readcmd  Variable Documentation	29 29 29 30 30 30 34 34
	8.2 8.3 8.4 8.5	READI ROADI src/ckb 8.5.1 8.5.2 8.5.3	DN.md File Reference  ME.md File Reference  MAP.md File Reference  -daemon/command.c File Reference  Macro Definition Documentation  8.5.1.1 TRY_WITH_RESET  Function Documentation  8.5.2.1 readcmd  Variable Documentation  8.5.3.1 cmd_strings	29 29 29 30 30 30 30 34
	8.2 8.3 8.4 8.5	READI ROADI src/ckb 8.5.1 8.5.2 8.5.3 src/ckb	DN.md File Reference  ME.md File Reference  MAP.md File Reference  -daemon/command.c File Reference  Macro Definition Documentation  8.5.1.1 TRY_WITH_RESET  Function Documentation  8.5.2.1 readcmd  Variable Documentation  8.5.3.1 cmd_strings  -daemon/command.h File Reference	29 29 29 30 30 30 34 34 34
	8.2 8.3 8.4 8.5	READI ROADI src/ckb 8.5.1 8.5.2 8.5.3 src/ckb	DN.md File Reference  ME.md File Reference  MAP.md File Reference  -daemon/command.c File Reference  Macro Definition Documentation  8.5.1.1 TRY_WITH_RESET  Function Documentation  8.5.2.1 readcmd  Variable Documentation  8.5.3.1 cmd_strings  -daemon/command.h File Reference  Data Structure Documentation	29 29 29 30 30 30 34 34 34 35
	8.2 8.3 8.4 8.5	DAEMI READI ROADI src/ckb 8.5.1 8.5.2 8.5.3 src/ckb 8.6.1	ME.md File Reference  MAP.md File Reference  -daemon/command.c File Reference  Macro Definition Documentation  8.5.1.1 TRY_WITH_RESET  Function Documentation  8.5.2.1 readcmd  Variable Documentation  8.5.3.1 cmd_strings  -daemon/command.h File Reference  Data Structure Documentation  8.6.1.1 union devcmd	29 29 29 30 30 30 34 34 35 35
	8.2 8.3 8.4 8.5	DAEMI READI ROADI src/ckb 8.5.1 8.5.2 8.5.3 src/ckb 8.6.1	ME.md File Reference  MAP.md File Reference  -daemon/command.c File Reference  Macro Definition Documentation  8.5.1.1 TRY_WITH_RESET  Function Documentation  8.5.2.1 readcmd  Variable Documentation  8.5.3.1 cmd_strings  -daemon/command.h File Reference  Data Structure Documentation  8.6.1.1 union devcmd  Macro Definition Documentation	29 29 29 30 30 30 34 34 35 35 36
	8.2 8.3 8.4 8.5	DAEMI READI ROADI src/ckb 8.5.1 8.5.2 8.5.3 src/ckb 8.6.1	ME.md File Reference  ME.md File Reference  MAP.md File Reference  -daemon/command.c File Reference  Macro Definition Documentation  8.5.1.1 TRY_WITH_RESET  Function Documentation  8.5.2.1 readcmd  Variable Documentation  8.5.3.1 cmd_strings  -daemon/command.h File Reference  Data Structure Documentation  8.6.1.1 union devcmd  Macro Definition Documentation  8.6.2.1 CMD_COUNT	29 29 29 30 30 30 34 34 35 35 36

CONTENTS

		8.6.3.1	cmdhandler	37
		8.6.3.2	cmdhandler_io	37
		8.6.3.3	cmdhandler_mac	37
		8.6.3.4	devcmd	37
	8.6.4	Enumera	tion Type Documentation	37
		8.6.4.1	cmd	37
	8.6.5	Function	Documentation	39
		8.6.5.1	readcmd	39
8.7	src/ckb	-daemon/d	device.c File Reference	43
	8.7.1	Function	Documentation	44
		8.7.1.1	_start_dev	44
		8.7.1.2	start_dev	45
	8.7.2	Variable	Documentation	45
		8.7.2.1	devlistmutex	45
		8.7.2.2	devmutex	45
		8.7.2.3	hwload_mode	45
		8.7.2.4	inputmutex	46
		8.7.2.5	keyboard	46
8.8	src/ckb	o-daemon/o	device.h File Reference	46
	8.8.1	Macro De	efinition Documentation	47
		8.8.1.1	ACT_LIGHT	47
		8.8.1.2	ACT_LOCK	47
		8.8.1.3	ACT_M1	47
		8.8.1.4	ACT_M2	47
		8.8.1.5	ACT_M3	47
		8.8.1.6	ACT_MR_RING	47
		8.8.1.7	ACT_NEXT	48
		8.8.1.8	ACT_NEXT_NOWRAP	48
		8.8.1.9	DEV_MAX	48
		8.8.1.10	dmutex	48
		8.8.1.11	imutex	48
		8.8.1.12	IN_CORSAIR	48
		8.8.1.13	IN_HID	48
		8.8.1.14	IS_CONNECTED	48
		8.8.1.15	setactive	48
	8.8.2	Function	Documentation	49
		8.8.2.1	cmd_active_kb	49
		8.8.2.2	cmd_active_mouse	49
		8.8.2.3	cmd_idle_kb	49
		8.8.2.4	cmd_idle_mouse	50

vi CONTENTS

		8.8.2.5	cmd_pollrate	50
		8.8.2.6	setactive_kb	50
		8.8.2.7	setactive_mouse	52
		8.8.2.8	setmodeindex_nrgb	53
		8.8.2.9	start_dev	54
		8.8.2.10	start_kb_nrgb	54
	8.8.3	Variable I	Documentation	54
		8.8.3.1	devmutex	54
		8.8.3.2	inputmutex	55
		8.8.3.3	keyboard	55
8.9	src/ckb	-daemon/d	device_keyboard.c File Reference	55
	8.9.1	Function	Documentation	55
		8.9.1.1	cmd_active_kb	55
		8.9.1.2	cmd_idle_kb	56
		8.9.1.3	setactive_kb	56
		8.9.1.4	setmodeindex_nrgb	58
		8.9.1.5	start_kb_nrgb	58
8.10	src/ckb	-daemon/d	device_mouse.c File Reference	59
	8.10.1	Function	Documentation	59
		8.10.1.1	cmd_active_mouse	59
		8.10.1.2	cmd_idle_mouse	60
		8.10.1.3	cmd_pollrate	60
		8.10.1.4	setactive_mouse	60
8.11	src/ckb	-daemon/d	device_vtable.c File Reference	62
	8.11.1	Function	Documentation	62
		8.11.1.1	cmd_io_none	62
		8.11.1.2	cmd_macro_none	63
		8.11.1.3	cmd_none	63
		8.11.1.4	int1_int_none	63
		8.11.1.5	int1_void_none	63
		8.11.1.6	loadprofile_none	63
	8.11.2	Variable I	Documentation	63
		8.11.2.1	vtable_keyboard	63
		8.11.2.2	vtable_keyboard_nonrgb	63
		8.11.2.3	vtable_mouse	64
8.12			devnode.c File Reference	64
	8.12.1	Data Stru	cture Documentation	65
		8.12.1.1	struct _readlines_ctx	65
	8.12.2	Macro De	efinition Documentation	65
		8.12.2.1	MAX_BUFFER	65

CONTENTS vii

	8.12.2.2 S_GID_READ	66
8.12.3	Function Documentation	66
	8.12.3.1 _mkdevpath	66
	8.12.3.2 _mknotifynode	68
	8.12.3.3 _rmnotifynode	69
	8.12.3.4 _updateconnected	69
	8.12.3.5 mkdevpath	70
	8.12.3.6 mkfwnode	71
	8.12.3.7 mknotifynode	71
	8.12.3.8 readlines	72
	8.12.3.9 readlines_ctx_free	73
	8.12.3.10 readlines_ctx_init	73
	8.12.3.11 rm_recursive	74
	8.12.3.12 rmdevpath	74
	8.12.3.13 rmnotifynode	75
	8.12.3.14 updateconnected	76
8.12.4	Variable Documentation	76
	8.12.4.1 devpath	76
	8.12.4.2 gid	76
8.13 src/ckt	b-daemon/devnode.h File Reference	76
8.13.1	Macro Definition Documentation	78
	8.13.1.1 S_CUSTOM	78
	8.13.1.2 S_CUSTOM_R	78
	8.13.1.3 S_READ	78
	8.13.1.4 S_READDIR	78
	8.13.1.5 S_READWRITE	78
8.13.2	Typedef Documentation	78
	8.13.2.1 readlines_ctx	78
8.13.3	Function Documentation	78
	8.13.3.1 mkdevpath	78
	8.13.3.2 mkfwnode	79
	8.13.3.3 mknotifynode	80
	8.13.3.4 readlines	80
	8.13.3.5 readlines_ctx_free	81
	8.13.3.6 readlines_ctx_init	82
	8.13.3.7 rmdevpath	82
	8.13.3.8 rmnotifynode	83
	8.13.3.9 updateconnected	84
8.13.4	Variable Documentation	84
	8.13.4.1 devpath	84

viii CONTENTS

	8.13.4.2 gid	34
8.14 src/ck	b-daemon/dpi.c File Reference	34
8.14.1	Function Documentation	35
	8.14.1.1 cmd_dpi	35
	8.14.1.2 cmd_dpisel	36
	8.14.1.3 cmd_lift	36
	8.14.1.4 cmd_snap	36
	8.14.1.5 loaddpi	36
	8.14.1.6 printdpi	37
	8.14.1.7 savedpi	38
	8.14.1.8 updatedpi	39
8.15 src/ck	b-daemon/dpi.h File Reference	39
8.15.1	Function Documentation	90
	8.15.1.1 cmd_dpi	90
	8.15.1.2 cmd_dpisel	91
	8.15.1.3 cmd_lift	91
	8.15.1.4 cmd_snap	91
	8.15.1.5 loaddpi	92
	8.15.1.6 printdpi	93
	8.15.1.7 savedpi	93
	8.15.1.8 updatedpi	94
8.16 src/ck	b-daemon/extra_mac.c File Reference	94
8.17 src/ck	b-daemon/firmware.c File Reference	95
8.17.1	Macro Definition Documentation	96
	8.17.1.1 FW_MAXSIZE	96
	8.17.1.2 FW_NOFILE	96
	8.17.1.3 FW_OK	96
	8.17.1.4 FW_USBFAIL	96
	8.17.1.5 FW_WRONGDEV	96
8.17.2	Prunction Documentation	96
	8.17.2.1 cmd_fwupdate	96
	8.17.2.2 fwupdate	97
	8.17.2.3 getfwversion	99
8.18 src/ck	b-daemon/firmware.h File Reference	00
8.18.1	Function Documentation	00
	8.18.1.1 cmd_fwupdate	00
	8.18.1.2 getfwversion	)1
8.19 src/ck	b-daemon/includes.h File Reference	)2
8.19.1	Macro Definition Documentation	)4
	8.19.1.1FILE_NOPATH	)4

CONTENTS

		8.19.1.2 ckb_err
		8.19.1.3 ckb_err_fn
		8.19.1.4 ckb_err_nofile
		8.19.1.5 ckb_fatal
		8.19.1.6 ckb_fatal_fn
		8.19.1.7 ckb_fatal_nofile
		8.19.1.8 ckb_info
		8.19.1.9 ckb_info_fn
		8.19.1.10 ckb_info_nofile
		8.19.1.11 ckb_s_err
		8.19.1.12 ckb_s_out
		8.19.1.13 ckb_warn
		8.19.1.14 ckb_warn_fn
		8.19.1.15 ckb_warn_nofile
		8.19.1.16 INDEX_OF
		8.19.1.17 timespec_eq
		8.19.1.18 timespec_ge
		8.19.1.19 timespec_gt
		8.19.1.20 timespec_le
		8.19.1.21 timespec_lt
	8.19.2	Typedef Documentation
		8.19.2.1 uchar
		8.19.2.2 ushort
	8.19.3	Function Documentation
		8.19.3.1 timespec_add
8.20	src/ckb	-daemon/input.c File Reference
	8.20.1	Macro Definition Documentation
		8.20.1.1 IS_WHEEL
	8.20.2	Function Documentation
		8.20.2.1 _cmd_macro
		8.20.2.2 cmd_bind
		8.20.2.3 cmd_macro
		8.20.2.4 cmd_rebind
		8.20.2.5 cmd_unbind
		8.20.2.6 freebind
		8.20.2.7 initbind
		8.20.2.8 inputupdate
		8.20.2.9 inputupdate_keys
		8.20.2.10 macromask
		8.20.2.11 updateindicators_kb

X CONTENTS

8.21	src/ckb	daemon/input.h File Reference
	8.21.1	Macro Definition Documentation
		8.21.1.1 IS_MOD
	8.21.2	Function Documentation
		8.21.2.1 cmd_bind
		8.21.2.2 cmd_macro
		8.21.2.3 cmd_rebind
		8.21.2.4 cmd_unbind
		8.21.2.5 freebind
		8.21.2.6 initbind
		8.21.2.7 inputupdate
		8.21.2.8 os_inputclose
		8.21.2.9 os_inputopen
		8.21.2.10 os_keypress
		8.21.2.11 os_mousemove
		8.21.2.12 os_setupindicators
		8.21.2.13 updateindicators_kb
8.22	src/ckb	daemon/input_linux.c File Reference
	8.22.1	Function Documentation
		8.22.1.1 _ledthread
		8.22.1.2 isync
		8.22.1.3 os_inputclose
		8.22.1.4 os_inputopen
		8.22.1.5 os_keypress
		8.22.1.6 os_mousemove
		8.22.1.7 os_setupindicators
		8.22.1.8 uinputopen
8.23	src/ckb	daemon/input_mac.c File Reference
8.24	src/ckb	daemon/input_mac_mouse.c File Reference
8.25	src/ckb	daemon/keymap.c File Reference
	8.25.1	Macro Definition Documentation
		8.25.1.1 BUTTON_HID_COUNT
	8.25.2	Function Documentation
		8.25.2.1 corsair_kbcopy
		8.25.2.2 corsair_mousecopy
		8.25.2.3 hid_kb_translate
		8.25.2.4 hid_mouse_translate
	8.25.3	Variable Documentation
		8.25.3.1 keymap
8.26	src/ckb	daemon/keymap.h File Reference

CONTENTS xi

	8.26.1	Data Structure Documentation
		8.26.1.1 struct key
	8.26.2	Macro Definition Documentation
		8.26.2.1 BTN_WHEELDOWN
		8.26.2.2 BTN_WHEELUP
		8.26.2.3 KEY_BACKSLASH_ISO
		8.26.2.4 KEY_CORSAIR
		8.26.2.5 KEY_NONE
		8.26.2.6 KEY_UNBOUND
		8.26.2.7 LED_DPI
		8.26.2.8 LED_MOUSE
		8.26.2.9 MOUSE_BUTTON_FIRST
		8.26.2.10 MOUSE_EXTRA_FIRST
		8.26.2.11 N_BUTTONS_EXTENDED
		8.26.2.12 N_BUTTONS_HW
		8.26.2.13 N_KEY_ZONES
		8.26.2.14 N_KEYBYTES_EXTENDED
		8.26.2.15 N_KEYBYTES_HW
		8.26.2.16 N_KEYBYTES_INPUT
		8.26.2.17 N_KEYS_EXTENDED
		8.26.2.18 N_KEYS_EXTRA
		8.26.2.19 N_KEYS_HW
		8.26.2.20 N_KEYS_INPUT
		8.26.2.21 N_MOUSE_ZONES
		8.26.2.22 N_MOUSE_ZONES_EXTENDED
		8.26.2.23 SCAN_KBD
		8.26.2.24 SCAN_MOUSE
		8.26.2.25 SCAN_SILENT
	8.26.3	Function Documentation
		8.26.3.1 corsair_kbcopy
		8.26.3.2 corsair_mousecopy
		8.26.3.3 hid_kb_translate
		8.26.3.4 hid_mouse_translate
	8.26.4	Variable Documentation
		8.26.4.1 keymap
8.27	src/ckb-	-daemon/keymap_mac.h File Reference
8.28	src/ckb-	-daemon/led.c File Reference
	8.28.1	Function Documentation
		8.28.1.1 cmd_iauto
		8.28.1.2 cmd_inotify

xii CONTENTS

	8.28.1.3 cmd_ioff
	8.28.1.4 cmd_ion
	8.28.1.5 cmd_rgb
	8.28.1.6 has_key
	8.28.1.7 iselect
	8.28.1.8 printrgb
8.29 src/ckk	o-daemon/led.h File Reference
8.29.1	Function Documentation
	8.29.1.1 cmd_iauto
	8.29.1.2 cmd_inotify
	8.29.1.3 cmd_ioff
	8.29.1.4 cmd_ion
	8.29.1.5 cmd_rgb
	8.29.1.6 loadrgb_kb
	8.29.1.7 loadrgb_mouse
	8.29.1.8 printrgb
	8.29.1.9 savergb_kb
	8.29.1.10 savergb_mouse
	8.29.1.11 updatergb_kb
	8.29.1.12 updatergb_mouse
8.30 src/ckb	o-daemon/led_keyboard.c File Reference
8.30.1	Macro Definition Documentation
	8.30.1.1 BR1
	8.30.1.2 BR2
	8.30.1.3 BR4
	8.30.1.4 O0
	8.30.1.5 O1
	8.30.1.6 O2
	8.30.1.7 O3
	8.30.1.8 O4
	8.30.1.9 O5
	8.30.1.10 O6
	8.30.1.11 O7
	8.30.1.12 O8
8.30.2	Function Documentation
	8.30.2.1 loadrgb_kb
	8.30.2.2 makergb_512
	8.30.2.3 makergb_full
	8.30.2.4 ordered8to3
	8.30.2.5 quantize8to3

CONTENTS xiii

		8.30.2.6	rgbcmp	189
		8.30.2.7	savergb_kb	190
		8.30.2.8	updatergb_kb	191
	8.30.3	Variable I	Documentation	192
		8.30.3.1	bit_reverse_table	193
8.31	src/ckb	-daemon/l	ed_mouse.c File Reference	194
	8.31.1	Function	Documentation	194
		8.31.1.1	isblack	194
		8.31.1.2	loadrgb_mouse	195
		8.31.1.3	rgbcmp	195
		8.31.1.4	savergb_mouse	196
		8.31.1.5	updatergb_mouse	196
8.32	src/ckb	-daemon/r	main.c File Reference	197
	8.32.1	Function	Documentation	198
		8.32.1.1	localecase	198
		8.32.1.2	main	199
		8.32.1.3	quit	201
		8.32.1.4	quitWithLock	202
		8.32.1.5	restart	203
		8.32.1.6	sighandler	204
		8.32.1.7	sighandler2	205
		8.32.1.8	timespec_add	205
	8.32.2	Variable I	Documentation	206
		8.32.2.1	features_mask	206
		8.32.2.2	hwload_mode	206
		8.32.2.3	main_ac	206
		8.32.2.4	main_av	206
		8.32.2.5	reset_stop	206
8.33	src/ckb	-daemon/r	notify.c File Reference	206
	8.33.1	Macro De	efinition Documentation	207
		8.33.1.1	HW_STANDARD	207
		8.33.1.2	HWMODE_OR_RETURN	207
	8.33.2	Function	Documentation	207
		8.33.2.1	_cmd_get	207
		8.33.2.2	cmd_get	210
		8.33.2.3	cmd_notify	211
		8.33.2.4	cmd_restart	211
		8.33.2.5	nprintf	212
		8.33.2.6	nprintind	213
		8.33.2.7	nprintkey	213

XIV

		8.33.2.8	restart .			 	 	 	 	 	 	 214
8.34	src/ckb	-daemon/r	notify.h File	Referen	ce	 	 	 	 	 	 	 215
	8.34.1	Function	Document	ation .		 	 	 	 	 	 	 216
		8.34.1.1	cmd_get			 	 	 	 	 	 	 216
		8.34.1.2	cmd_notif	y		 	 	 	 	 	 	 217
		8.34.1.3	cmd_resta	art		 	 	 	 	 	 	 217
		8.34.1.4	nprintf .			 	 	 	 	 	 	 217
		8.34.1.5	nprintind			 	 	 	 	 	 	 218
		8.34.1.6	nprintkey			 	 	 	 	 	 	 219
8.35	src/ckb	-daemon/d	s.h File Re	eference		 	 	 	 	 	 	 219
	8.35.1	Macro De	efinition Do	cumenta	tion .	 	 	 	 	 	 	 220
		8.35.1.1	_DEFAUL	.T_SOUI	RCE .	 	 	 	 	 	 	 220
		8.35.1.2	_GNU_S	OURCE		 	 	 	 	 	 	 220
		8.35.1.3	euid_gua	d_start		 	 	 	 	 	 	 220
		8.35.1.4	euid_gua	rd_stop		 	 	 	 	 	 	 220
		8.35.1.5	UINPUT_	VERSIO	Ν	 	 	 	 	 	 	 221
8.36	src/ckb	-daemon/p	orofile.c File	Refere	nce .	 	 	 	 	 	 	 221
	8.36.1	Function	Document	ation .		 	 	 	 	 	 	 222
		8.36.1.1	_freeprofi	le		 	 	 	 	 	 	 222
		8.36.1.2	allocprofil	е		 	 	 	 	 	 	 222
		8.36.1.3	cmd_eras	е		 	 	 	 	 	 	 223
		8.36.1.4	cmd_eras	eprofile		 	 	 	 	 	 	 224
		8.36.1.5	cmd_id .			 	 	 	 	 	 	 224
		8.36.1.6	cmd_nam	е		 	 	 	 	 	 	 225
		8.36.1.7	cmd_prof	leid		 	 	 	 	 	 	 225
		8.36.1.8	cmd_prof	lename		 	 	 	 	 	 	 226
		8.36.1.9	freemode			 	 	 	 	 	 	 226
		8.36.1.10	freeprofile			 	 	 	 	 	 	 227
		8.36.1.11	gethwmod	dename		 	 	 	 	 	 	 227
		8.36.1.12	gethwpro	ilename		 	 	 	 	 	 	 228
		8.36.1.13	getid			 	 	 	 	 	 	 229
		8.36.1.14	getmoder	ame .		 	 	 	 	 	 	 229
		8.36.1.15	getprofile	name .		 	 	 	 	 	 	 230
		8.36.1.16	hwtonativ	е		 	 	 	 	 	 	 231
		8.36.1.17	initmode			 	 	 	 	 	 	 231
		8.36.1.18	loadprofile	·		 	 	 	 	 	 	 232
		8.36.1.19	nativetoh	<b>v</b>		 	 	 	 	 	 	 232
		8.36.1.20	printname			 	 	 	 	 	 	 233
		8.36.1.21	setid			 	 	 	 	 	 	 234
		8.36.1.22	u16dec .			 	 	 	 	 	 	 234

CONTENTS xv

		8.36.1.23 u16enc
		8.36.1.24 urldecode2
		8.36.1.25 urlencode2
	8.36.2	Variable Documentation
		8.36.2.1 utf16to8
		8.36.2.2 utf8to16
8.37	src/ckb	daemon/profile.h File Reference
	8.37.1	Macro Definition Documentation
		8.37.1.1 hwloadprofile
	8.37.2	Function Documentation
		8.37.2.1 allocprofile
		8.37.2.2 cmd_erase
		8.37.2.3 cmd_eraseprofile
		8.37.2.4 cmd_hwload_kb
		8.37.2.5 cmd_hwload_mouse
		8.37.2.6 cmd_hwsave_kb
		8.37.2.7 cmd_hwsave_mouse
		8.37.2.8 cmd_id
		8.37.2.9 cmd_name
		8.37.2.10 cmd_profileid
		8.37.2.11 cmd_profilename
		8.37.2.12 freeprofile
		8.37.2.13 gethwmodename
		8.37.2.14 gethwprofilename
		8.37.2.15 getid
		8.37.2.16 getmodename
		8.37.2.17 getprofilename
		8.37.2.18 hwtonative
		8.37.2.19 loadprofile
		8.37.2.20 nativetohw
		8.37.2.21 setid
8.38	src/ckb	daemon/profile_keyboard.c File Reference
	8.38.1	Function Documentation
		8.38.1.1 cmd_hwload_kb
		8.38.1.2 cmd_hwsave_kb
		8.38.1.3 hwloadmode
8.39	src/ckb	daemon/profile_mouse.c File Reference
	8.39.1	Function Documentation
		8.39.1.1 cmd_hwload_mouse
		8.39.1.2 cmd_hwsave_mouse

xvi CONTENTS

8.40	src/ckb	-daemon/st	ructures.h File Reference	 	 257
	8.40.1	Data Struc	cture Documentation	 	 259
		8.40.1.1	struct usbid	 	259
		8.40.1.2	struct macroaction	 	260
		8.40.1.3	struct keymacro	 	261
		8.40.1.4	struct binding	 	262
		8.40.1.5	struct dpiset	 	 263
		8.40.1.6	struct lighting	 	 264
		8.40.1.7	struct usbmode	 	 265
		8.40.1.8	struct usbprofile	 	 266
		8.40.1.9	struct hwprofile	 	267
		8.40.1.10	struct usbinput	 	 268
		8.40.1.11	struct usbdevice	 	 269
	8.40.2	Macro Det	finition Documentation	 	 271
		8.40.2.1	CLEAR_KEYBIT	 	271
		8.40.2.2	DPI_COUNT	 	 271
		8.40.2.3	FEAT_ADJRATE	 	 271
		8.40.2.4	FEAT_ANSI	 	 271
		8.40.2.5	FEAT_BIND	 	 271
		8.40.2.6	FEAT_COMMON	 	 271
		8.40.2.7	FEAT_FWUPDATE	 	 271
		8.40.2.8	FEAT_FWVERSION	 	 272
		8.40.2.9	FEAT_HWLOAD	 	272
		8.40.2.10	FEAT_ISO	 	 272
		8.40.2.11	FEAT_LMASK	 	 272
		8.40.2.12	FEAT_MONOCHROME	 	 272
		8.40.2.13	FEAT_MOUSEACCEL	 	272
		8.40.2.14	FEAT_NOTIFY	 	272
		8.40.2.15	FEAT_POLLRATE	 	 272
		8.40.2.16	FEAT_RGB	 	 272
		8.40.2.17	FEAT_STD_NRGB	 	 273
		8.40.2.18	FEAT_STD_RGB	 	273
		8.40.2.19	HAS_ANY_FEATURE	 	273
		8.40.2.20	HAS_FEATURES	 	 273
		8.40.2.21	HWMODE_K70	 	 273
		8.40.2.22	HWMODE_K95	 	 273
		8.40.2.23	HWMODE_MAX	 	 273
		8.40.2.24	I_CAPS	 	 273
		8.40.2.25	I_NUM	 	 273
		8.40.2.26	I_SCROLL	 	 273

CONTENTS xvii

	8.40.2.27	IFACE_MAX	<b>(</b>												274
	8.40.2.28	KB_NAME_	LEN												274
	8.40.2.29	LIFT_MAX													274
	8.40.2.30	LIFT_MIN													274
	8.40.2.31	MACRO_M/	<b>чх</b>												274
	8.40.2.32	MD_NAME_	LEN												274
	8.40.2.33	MODE_COL	JNT												274
	8.40.2.34	MSG_SIZE													274
	8.40.2.35	NEEDS_FW	_UPDA	TE											274
	8.40.2.36	OUTFIFO_N	л <b>АХ</b>												275
	8.40.2.37	PR_NAME_	LEN												275
	8.40.2.38	SCROLL_A	CCELEF	RATED											275
	8.40.2.39	SCROLL_M	<b>AX</b>												275
	8.40.2.40	SCROLL_M	IN												275
	8.40.2.41	SERIAL_LE	Ν												275
	8.40.2.42	SET_KEYB	т												275
.40.3	Variable D	ocumentatio	n												275
	8.40.3.1	vtable_keyb	oard												275
	8.40.3.2	vtable_keyb	oard_noi	nrgb .											275
	8.40.3.3	vtable_mous	se												276
rc/ckb-	daemon/u	sb.c File Ref	erence .												276
.41.1	Function [	Documentation	on												277
	8.41.1.1	_resetusb													277
	8.41.1.2	_setupusb													277
	8.41.1.3	_usbrecv .													279
	8.41.1.4	_usbsend .													280
	8.41.1.5	closeusb .													281
	8.41.1.6	devmain .													282
	8.41.1.7	get_vtable													283
	8.41.1.8	product_str													283
	8.41.1.9	revertusb .													284
	8.41.1.10	setupusb .													285
	8.41.1.11	usb_tryrese	t												285
	8.41.1.12	vendor_str													286
.41.2	Variable D	ocumentatio	n												286
	8.41.2.1	features_ma	ısk												286
	8.41.2.2	hwload_mod	de												286
	8.41.2.3	reset_stop													287
	8.41.2.4	usbmutex .													287
rc/ckb-	daemon/u	sb.h File Ref	erence .												287
	.40.3 rc/ckb- .41.1	8.40.2.28 8.40.2.30 8.40.2.31 8.40.2.32 8.40.2.33 8.40.2.34 8.40.2.35 8.40.2.36 8.40.2.37 8.40.2.38 8.40.2.39 8.40.2.40 8.40.2.41 8.40.2.42 40.3 Variable D 8.40.3.1 8.40.3.2 8.40.3.3 rc/ckb-daemon/u- 8.41.1.1 8.41.1.2 8.41.1.3 8.41.1.4 8.41.1.5 8.41.1.6 8.41.1.7 8.41.1.8 8.41.1.9 8.41.1.10 8.41.1.11 8.41.1.12 41.2 Variable D 8.41.1.10 8.41.1.11	8.40.2.28 KB_NAME_ 8.40.2.29 LIFT_MAX 8.40.2.30 LIFT_MIN 8.40.2.31 MACRO_MA 8.40.2.32 MD_NAME_ 8.40.2.33 MODE_COU 8.40.2.34 MSG_SIZE 8.40.2.35 NEEDS_FW 8.40.2.36 OUTFIFO_M 8.40.2.37 PR_NAME_ 8.40.2.38 SCROLL_AG 8.40.2.39 SCROLL_M 8.40.2.40 SCROLL_M 8.40.2.41 SERIAL_LEI 8.40.2.42 SET_KEYBI 8.40.3.1 vtable_keybo 8.40.3.2 vtable_keybo 8.40.3.3 vtable_mous rc/ckb-daemon/usb.c File Refo 8.41.1.1 _resetusb 8.41.1.2 _setupusb 8.41.1.2 _setupusb 8.41.1.3 _usbrecv 8.41.1.4 _usbsend 8.41.1.5 closeusb 8.41.1.5 closeusb 8.41.1.6 devmain 8.41.1.7 get_vtable 8.41.1.7 get_vtable 8.41.1.8 product_str 8.41.1.9 revertusb 8.41.1.10 setupusb 8.41.1.11 usb_tryreset 8.41.1.11 usb_tryreset 8.41.1.11 resetusb 8.41.1.11 usb_tryreset 8.41.1.11 resetusb 8.41.1.11 usb_tryreset 8.41.1.11 usb_tryreset 8.41.1.12 vendor_str 8.41.2.1 features_ma 8.41.2.2 hwload_mod 8.41.2.3 reset_stop 8.41.2.4 usbmutex.	8.40.2.28 KB_NAME_LEN	8.40.2.28 KB_NAME_LEN 8.40.2.29 LIFT_MAX 8.40.2.30 LIFT_MIN 8.40.2.31 MACRO_MAX 8.40.2.32 MD_NAME_LEN 8.40.2.33 MODE_COUNT 8.40.2.34 MSG_SIZE 8.40.2.35 NEEDS_FW_UPDATE 8.40.2.36 OUTFIFO_MAX 8.40.2.37 PR_NAME_LEN 8.40.2.38 SCROLL_ACCELERATED 8.40.2.39 SCROLL_MIN 8.40.2.40 SCROLL_MIN 8.40.2.41 SERIAL_LEN 8.40.2.42 SET_KEYBIT 40.3 Variable Documentation 8.40.3.1 vtable_keyboard 8.40.3.2 vtable_keyboard_nonrgb 8.40.3.3 vtable_mouse 6000000000000000000000000000000000000	8.40.2.28 KB_NAME_LEN 8.40.2.29 LIFT_MAX 8.40.2.30 LIFT_MIN 8.40.2.31 MACRO_MAX 8.40.2.32 MD_NAME_LEN 8.40.2.33 MODE_COUNT 8.40.2.34 MSG_SIZE 8.40.2.35 NEEDS_FW_UPDATE 8.40.2.36 OUTFIFO_MAX 8.40.2.37 PR_NAME_LEN 8.40.2.39 SCROLL_ACCELERATED 8.40.2.39 SCROLL_MIN 8.40.2.41 SERIAL_LEN 8.40.2.42 SET_KEYBIT 8.40.3.1 vtable_keyboard 8.40.3.2 vtable_keyboard_nonrgb 8.40.3.3 vtable_mouse 8.40.3.3 vtable_mouse 8.41.1.1 _resetusb 8.41.1.2 _setupusb 8.41.1.3 _usbrecv 8.41.1.4 _usbsend 8.41.1.5 closeusb 8.41.1.6 devmain 8.41.1.7 get_vtable 8.41.1.9 revertusb 8.41.1.10 setupusb 8.41.1.10 revertusb 8.41.1.11 usb_tryreset 8.41.1.11 usb_tryreset 8.41.1.11 vsetupusb 8.41.1.11 resetusb 8.41.1.11 revertusb 8.41.1.12 vendor_str 8.41.1.11 usb_tryreset 8.41.1.11 usb_tryreset 8.41.2.1 features_mask 8.41.2.2 hwload_mode 8.41.2.3 reset_stop 8.41.2.4 usbmutex	8.40.2.28 KB_NAME_LEN 8.40.2.29 LIFT_MAX 8.40.2.30 LIFT_MIN 8.40.2.31 MACRO_MAX 8.40.2.32 MD_NAME_LEN 8.40.2.33 MODE_COUNT 8.40.2.35 NEEDS_FW_UPDATE 8.40.2.36 OUTFIFO_MAX 8.40.2.37 PR_NAME_LEN 8.40.2.39 SCROLL_ACCELERATED 8.40.2.39 SCROLL_MAX 8.40.2.40 SCROLL_MIN 8.40.2.41 SERIAL_LEN 8.40.2.42 SET_KEYBIT 40.3 Variable Documentation 8.40.3.1 vtable_keyboard 8.40.3.2 vtable_keyboard 8.40.3.3 vtable_mouse rc/ckb-daemon/usb.c File Reference 41.1 Function Documentation 8.41.1.1 _resetusb 8.41.1.2 _setupusb 8.41.1.3 _usbrecv 8.41.1.4 _usbsend 8.41.1.5 closeusb 8.41.1.6 devmain 8.41.1.7 get_vtable 8.41.1.8 product_str 8.41.1.9 revertusb 8.41.1.10 setupusb 8.41.1.11 usb_tryreset 8.41.1.11 usb_tryreset 8.41.1.11 usb_tryreset 8.41.1.12 heatures_mask 8.41.2.2 hwload_mode 8.41.2.1 features_mask 8.41.2.2 hwload_mode 8.41.2.3 reset_stop 8.41.2.4 usbmutex	8.40.2.28 KB_NAME_LEN  8.40.2.29 LIFT_MAX  8.40.2.30 LIFT_MIN  8.40.2.31 MACRO_MAX  8.40.2.32 MD_NAME_LEN  8.40.2.33 MODE_COUNT  8.40.2.34 MSG_SIZE  8.40.2.35 NEEDS_FW_UPDATE  8.40.2.36 OUTFIFO_MAX  8.40.2.37 PR_NAME_LEN  8.40.2.39 SCROLL_ACCELERATED  8.40.2.39 SCROLL_MAX  8.40.2.40 SCROLL_MIN  8.40.2.41 SERIAL_LEN  8.40.2.42 SET_KEYBIT  40.3 Variable Documentation  8.40.3.1 vtable_keyboard  8.40.3.2 vtable_keyboard_nonrgb  8.40.3.3 vtable_mouse  **C/ckb-daemon/usb.c File Reference  4.1.1 Function Documentation  8.41.1.1 _resetusb  8.41.1.2 _setupusb  8.41.1.3 _usbrecv  8.41.1.4 _usbsend  8.41.1.5 closeusb  8.41.1.6 devmain  8.41.1.7 get_vtable  8.41.1.8 product_str  8.41.1.10 setupusb  8.41.1.11 usb_tryreset  8.41.1.11 usb_tryreset  8.41.1.11 usb_tryreset  8.41.1.12 readures_mask  8.41.2.1 features_mask  8.41.2.3 reset_stop  8.41.2.4 usbmutex	8.40.2.28 KB_NAME_LEN 8.40.2.30 LIFT_MIN 8.40.2.31 MACRO_MAX 8.40.2.32 MD_NAME_LEN 8.40.2.33 MODE_COUNT 8.40.2.34 MSG_SIZE 8.40.2.35 NEEDS_FW_UPDATE 8.40.2.36 OUTFIFO_MAX 8.40.2.37 PR_NAME_LEN 8.40.2.39 SCROLL_ACCELERATED 8.40.2.39 SCROLL_MAX 8.40.2.40 SCROLL_MIN 8.40.2.41 SERIAL_LEN 8.40.2.42 SET_KEYBIT 40.3 Variable Documentation 8.40.3.1 vtable_keyboard 8.40.3.2 vtable_keyboard_nonrgb 8.40.3.3 vtable_mouse **C/ckb-daemon/usb.c File Reference 4.1.1 Function Documentation 8.41.1.1 _resetusb 8.41.1.2 _setupusb 8.41.1.3 _usbrecv 8.41.1.4 _usbsend 8.41.1.5 closeusb 8.41.1.6 devmain 8.41.1.7 get_vtable 8.41.1.10 setupusb 8.41.1.11 usb_tryreset 8.41.1.11 usb_tryreset 8.41.1.11 usb_tryreset 8.41.1.11 vendor_str 4.12 Variable Documentation 8.41.2.1 features_mask 8.41.2.2 hwload_mode 8.41.2.3 reset_stop 8.41.2.3 reset_stop 8.41.2.4 usbmutex	8.40.2.28 KB_NAME_LEN 8.40.2.30 LIFT_MIN 8.40.2.31 MACRO_MAX 8.40.2.32 MD_NAME_LEN 8.40.2.33 MODE_COUNT 8.40.2.34 MSG_SIZE 8.40.2.35 NEEDS_FW_UPDATE 8.40.2.36 OUTFIFO_MAX 8.40.2.37 PR_NAME_LEN 8.40.2.39 SCROLL_MAX 8.40.2.39 SCROLL_MAX 8.40.2.41 SERIAL_LEN 8.40.2.41 SERIAL_LEN 8.40.2.42 SET_KEYBIT 40.3 Variable Documentation 8.40.3.1 vtable_keyboard 8.40.3.2 vtable_keyboard_nonrgb 8.40.3.3 vtable_mouse **C/ckb-daemon/usb.c File Reference 41.1 Function Documentation 8.41.1.1 _resetusb 8.41.1.2 _setupusb 8.41.1.3 _usbrecv 8.41.1.4 _usbsend 8.41.1.5 closeusb 8.41.1.6 devmain 8.41.1.7 get_vtable 8.41.1.10 setupusb 8.41.1.10 setupusb 8.41.1.10 setupusb 8.41.1.11 usb_tryreset 8.41.1.10 setupusb 8.41.1.11 usb_tryreset 8.41.1.11 vshordor_str 4.12 Variable Documentation 8.41.2.1 features_mask 8.41.2.2 hyload_mode 8.41.2.3 reset_stop 8.41.2.3 reset_stop 8.41.2.4 usbmutex.	8.40.2.28 KB_NAME_LEN 8.40.2.29 LIFT_MAX 8.40.2.30 LIFT_MIN 8.40.2.31 MACRO_MAX 8.40.2.32 MD_NAME_LEN 8.40.2.33 MODE_COUNT 8.40.2.34 MSG_SIZE 8.40.2.35 NEEDS_FW_UPDATE 8.40.2.36 OUTFIFO_MAX 8.40.2.37 PR_NAME_LEN 8.40.2.39 SCROLL_ACCELERATED 8.40.2.39 SCROLL_MAX 8.40.2.40 SCROLL_MIN 8.40.2.41 SERIAL_LEN 8.40.2.42 SET_KEYBIT 40.3 Variable Documentation 8.40.3.1 vtable_keyboard 8.40.3.2 vtable_keyboard 8.40.3.3 vtable_mouse 6000000000000000000000000000000000000	8.40.2.28 KB_NAME_LEN 8.40.2.29 LIFT_MAX 8.40.2.30 LIFT_MIN 8.40.2.31 MACRO_MAX 8.40.2.32 MD_NAME_LEN 8.40.2.33 MODE_COUNT 8.40.2.34 MSG_SIZE 8.40.2.35 NEEDS_FW_UPDATE 8.40.2.36 OUTFIFO_MAX 8.40.2.37 PR_NAME_LEN 8.40.2.38 SCROLL_ACCELERATED 8.40.2.39 SCROLL_MAX 8.40.2.40 SCROLL_MIN 8.40.2.41 SERIAL_LEN 8.40.2.42 SET_KEYBIT 40.3 Variable Documentation 8.40.3.1 vtable_keyboard 8.40.3.2 vtable_mouse 6000000000000000000000000000000000000	8.40.2.28 KB NAME LEN 8.40.2.29 LIFT_MAX 8.40.2.30 LIFT_MIN 8.40.2.31 MACRO_MAX 8.40.2.32 MD_NAME_LEN 8.40.2.33 MODE_COUNT 8.40.2.34 MSG_SIZE 8.40.2.35 NEEDS_FW_UPDATE 8.40.2.36 OUTFIFO_MAX 8.40.2.37 PR_NAME_LEN 8.40.2.39 SCROLL_ACCELERATED 8.40.2.39 SCROLL_ACCELERATED 8.40.2.39 SCROLL_MIN 8.40.2.41 SERIAL_LEN 8.40.2.42 SET_KEYBIT 40.3 Variable Documentation 8.40.3.1 vtable_keyboard 8.40.3.2 vtable_keyboard 8.40.3.2 vtable_keyboard 8.40.3.3 vtable_mouse ro/ckb-daemon/usb.c File Reference 41.1 Function Documentation 8.41.1.1 _resetusb 8.41.1.2 _setupusb 8.41.1.3 _usbrecv 8.41.1.4 _usbsend 8.41.1.5 closeusb 8.41.1.6 devmain 8.41.1.7 get_vtable 8.41.1.8 product_str 8.41.1.9 revertusb 8.41.1.10 setupusb 8.41.1.11 usb_tryreset 8.41.1.11 usb_tryreset 8.41.1.11 tusb_tryreset 8.41.1.11 teatures_mask 8.41.2 teatures_mask 8.41.2 ineat_resetusp 8.41.2.1 features_mask 8.41.2 tusbmutex	8.40.2.28 KB_NAME_LEN 8.40.2.30 LIFT_MIN 8.40.2.31 MACRO_MAX 8.40.2.32 MD_NAME_LEN 8.40.2.33 MODE_COUNT 8.40.2.34 MSG_SIZE 8.40.2.35 NEEDS_FW_UPDATE 8.40.2.36 OUTFIFO_MAX 8.40.2.37 PR_NAME_LEN 8.40.2.38 SCROLL_MCCELERATED 8.40.2.39 SCROLL_MAX 8.40.2.40 SCROLL_MIN 8.40.2.41 SERIAL_LEN 8.40.2.42 SET_KEYBIT 4.0.3 Variable Documentation 8.40.3.1 vtable_keyboard_nonrgb 8.40.3.2 vtable_keyboard_nonrgb 8.40.3.3 vtable_mouse **CrCrCrCrCrCrCrCrCrCrCrCrCrCrCrCrCrCrC	8.41.1.5 closeusb

xviii CONTENTS

8.42.1	Macro De	finition Documentation	 	 289
	8.42.1.1	DELAY_LONG	 	 289
	8.42.1.2	DELAY_MEDIUM	 	 289
	8.42.1.3	DELAY_SHORT	 	 290
	8.42.1.4	IS_FULLRANGE	 	 290
	8.42.1.5	IS_K65	 	 290
	8.42.1.6	IS_K70	 	 290
	8.42.1.7	IS_K95	 	 290
	8.42.1.8	IS_M65	 	 290
	8.42.1.9	IS_MONOCHROME	 	 290
	8.42.1.10	IS_MONOCHROME_DEV	 	 290
	8.42.1.11	IS_MOUSE	 	 290
	8.42.1.12	IS_MOUSE_DEV	 	 291
	8.42.1.13	IS_RGB	 	 291
	8.42.1.14	IS_RGB_DEV	 	 291
	8.42.1.15	IS_SABRE	 	 291
	8.42.1.16	IS_SCIMITAR	 	 291
	8.42.1.17	IS_STRAFE	 	 291
	8.42.1.18	NK95_HWOFF	 	 291
	8.42.1.19	NK95_HWON	 	 291
	8.42.1.20	NK95_M1	 	 292
	8.42.1.21	NK95_M2	 	 292
	8.42.1.22	NK95_M3	 	 292
	8.42.1.23	nk95cmd	 	 292
	8.42.1.24	P_K65	 	 292
	8.42.1.25	P_K65_LUX	 	 292
	8.42.1.26	P_K65_LUX_STR	 	 292
	8.42.1.27	P_K65_NRGB	 	 292
	8.42.1.28	P_K65_NRGB_STR	 	 292
	8.42.1.29	P_K65_RFIRE	 	 292
	8.42.1.30	P_K65_RFIRE_STR	 	 293
	8.42.1.31	P_K65_STR	 	 293
	8.42.1.32	P_K70	 	 293
	8.42.1.33	P_K70_LUX	 	 293
	8.42.1.34	P_K70_LUX_NRGB	 	 293
	8.42.1.35	P_K70_LUX_NRGB_STR	 	 293
	8.42.1.36	P_K70_LUX_STR	 	 293
	8.42.1.37	P_K70_NRGB	 	 293
	8.42.1.38	P_K70_NRGB_STR	 	 293
	8.42.1.39	P_K70_RFIRE	 	 293

CONTENTS xix

	8.42.1.40 P_K70_RFIRE_NRGB	94
	8.42.1.41 P_K70_RFIRE_NRGB_STR	94
	8.42.1.42 P_K70_RFIRE_STR	94
	8.42.1.43 P_K70_STR	94
	8.42.1.44 P_K95	94
	8.42.1.45 P_K95_NRGB	94
	8.42.1.46 P_K95_NRGB_STR	94
	8.42.1.47 P_K95_PLATINUM	94
	8.42.1.48 P_K95_PLATINUM_STR	94
	8.42.1.49 P_K95_STR	94
	8.42.1.50 P_M65	94
	8.42.1.51 P_M65_PRO	95
	8.42.1.52 P_M65_PRO_STR	95
	8.42.1.53 P_M65_STR	95
	8.42.1.54 P_SABRE_L	95
	8.42.1.55 P_SABRE_L_STR	95
	8.42.1.56 P_SABRE_N	
	8.42.1.57 P_SABRE_N_STR	95
	8.42.1.58 P_SABRE_O	95
	8.42.1.59 P_SABRE_O2	95
	8.42.1.60 P_SABRE_O2_STR	95
	8.42.1.61 P_SABRE_O_STR	
	8.42.1.62 P_SCIMITAR	96
	8.42.1.63 P_SCIMITAR_PRO	96
	8.42.1.64 P_SCIMITAR_PRO_STR	96
	8.42.1.65 P_SCIMITAR_STR	96
	8.42.1.66 P_STRAFE	
	8.42.1.67 P_STRAFE_NRGB	96
	8.42.1.68 P_STRAFE_NRGB_STR	96
	8.42.1.69 P_STRAFE_STR	96
	8.42.1.70 resetusb	96
	8.42.1.71 USB_DELAY_DEFAULT	96
	8.42.1.72 usbrecv	97
	8.42.1.73 usbsend	97
	8.42.1.74 V_CORSAIR	97
	8.42.1.75 V_CORSAIR_STR	97
8.42.2	Function Documentation	
	8.42.2.1 _nk95cmd	
	8.42.2.2 _resetusb	
	8.42.2.3 _usbrecv	98

CONTENTS

	8.42.2.4 _usbsend
	8.42.2.5 closeusb
	8.42.2.6 os_closeusb
	8.42.2.7 os_inputmain
	8.42.2.8 os_resetusb
	8.42.2.9 os_sendindicators
	8.42.2.10 os_setupusb
	8.42.2.11 os_usbrecv
	8.42.2.12 os_usbsend
	8.42.2.13 product_str
	8.42.2.14 revertusb
	8.42.2.15 setupusb
	8.42.2.16 usb_tryreset
	8.42.2.17 usbkill
	8.42.2.18 usbmain
	8.42.2.19 vendor_str
8.43 src/ckl	o-daemon/usb_linux.c File Reference
8.43.1	Data Structure Documentation
	8.43.1.1 struct_model
8.43.2	Macro Definition Documentation
	8.43.2.1 N_MODELS
	8.43.2.2 TEST_RESET
8.43.3	Function Documentation
	8.43.3.1 _nk95cmd
	8.43.3.2 os_closeusb
	8.43.3.3 os_inputmain
	8.43.3.4 os_resetusb
	8.43.3.5 os_sendindicators
	8.43.3.6 os_setupusb
	8.43.3.7 os_usbrecv
	8.43.3.8 os_usbsend
	8.43.3.9 strtrim
	8.43.3.10 udev_enum
	8.43.3.11 usb_add_device
	8.43.3.12 usb_rm_device
	8.43.3.13 usbadd
	8.43.3.14 usbclaim
	8.43.3.15 usbkill
	8.43.3.16 usbmain
	8.43.3.17 usbunclaim

CONTENTS xxi

Index																331
8.44 src/ckb	o-daemon/เ	usb_mac.c	File I	Refe	rend	ce .		 								 329
	8.43.4.5	usbthread						 								 329
	8.43.4.4	udevthrea	d					 								 329
	8.43.4.3	udev						 								 329
	8.43.4.2	models .						 								 328
	8.43.4.1	kbsyspath						 								 328
8.43.4	Variable I	Documenta	ion					 								 328

# **Chapter 1**

# ckb-next: RGB Driver for Linux and OS X

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

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

- 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"
- ckb testing repository (updated more frequently, but may be unstable)

# **Device Support**

### Keyboards

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

#### Mice

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

# **Linux Installation**

# Pre-made packages

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

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

#### Preparation

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

- **Ubuntu**: sudo apt-get install build-essential libudev-dev qt5-default zlib1g-dev libappindicator-dev
- Fedora: sudo dnf install zlib-devel qt5-qtbase-devel libgudev-devel libappindicator-de systemd-devel qcc-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 <code>CONFIG\_INPUT\_UINPUT</code> flag to be enabled. It is located in <code>Device Drivers -> Input Device Support -> Miscellaneous devices -> User level driver support. If you don't know what this means, you can ignore this.</code>

#### 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 thequickinstallscript and proceed to the service installation. The script will saySystem service: Upstart detectedorSystem 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 <code>xcode-select --install</code> 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 <code>Xcode -> Preferences -> Locations -> Command Line Tools</code> and select an SDK version. Afterwards install Qt5 from here: <a href="http://www.qt.io/download-open-source/">http://www.qt.io/download-open-source/</a>

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

### **Upgrading (binary)**

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

#### **Upgrading** (source)

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

#### Uninstalling

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

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

### **Usage**

The user interface is still a work in progress.

#### **Major features**

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

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

#### Roadmap

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

### **Troubleshooting**

#### Linux

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

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

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

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

If it still doesn't work, try replacing  $0 \times 20000408$  with  $0 \times 4$ . Note that this will cause the kernel driver to ignore the device(s) completely, so you need to ensure ckb-daemon is running at boot or else you'll have no input. This will not work if you are using full-disk encryption.

If you see **GLib** critical errors like "'GLib-GObject-CRITICAL \*\*: g\_type\_add\_interface\_static: assertion 'G\_TYPE\_IS\_INSTANTIATABLE (instance\_type)' failed "'read this Arch Linux thread and try different combinations from it. If it doesn't help, you might want get support from your distribution community and tell them you cannot solve the problem in this thread.

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

#### OS X/macOS

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

### General

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

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

#### Common issues:

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

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

#### Reporting issues

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

#### Known issues

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

# Contributing

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

# Contact us

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

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

# What happened to the original ckb

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

ckb-next:	<b>BCB</b>	Driver	for I	inuv	and	20	v
CKD-NEXI:	מנוח	Driver	ior i	ITHIX	ano	$\omega$	

# **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-daem 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.

10 Building ckb

#### 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 <math>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 <code>'service/launchd/com.ckb.daemon.plist'</code> to your computer's <code>/Library/LaunchDaemons</code> folder (you can get to it by pressing <code>Cmd+Shift+G</code> 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 <code>sudo ckb.app/Contents/Resources/ckb-daemon</code> to start the driver. Open <code>ckb.app</code> 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.

cbk l	Improv	ements	Roadma	ľ
-------	--------	--------	--------	---

# **Chapter 4**

# **DAEMON**

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

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

/dev/input/ckb0 contains the following files:

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

Other ckb\* devices contain the following:

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

### **Commands**

The /dev/input/ckb\*/cmd nodes accept input in the form of text commands. They are normally accessible to all users on the system (see Security section). Commands should be given in the following format: [mode < n>] command1 [parameter1] [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 > mode < n >

14 DAEMON

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.
- $\bullet$  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.

16 DAEMON

#### **Indicators**

The indicator LEDs (Num Lock, Caps Lock, Scroll Lock) are controlled with the  $\pm$  commands.

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

#### Binding keys

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

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

#### **Examples:**

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

## **Key macros**

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

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

### **Examples:**

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

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

#### **DPI** and mouse settings

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

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

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

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

#### Additional mouse settings:

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

#### **Notifications**

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

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

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

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

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

# Examples:

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

18 DAEMON

#### 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 <quid> <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.
- $\operatorname{get}$  : dpi returns a dpi command equivalent to the current DPI bank.
- get :dpisel returns a dpisel command for the currently-selected DPI stage.
- get :lift returns a lift command for the current lift height.
- get :snap returns the current angle snap status.
- get :hwdpi, get :hwdpisel, get :hwlift, and get :hwsnap return the same properties, but for the current hardware profile.
- get :keys and get :i return the current keypress status and indicator status, respectively. They will indicate all currently pressed keys and all currently active indicators, like key +enter and i +num.

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

#### Firmware updates

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

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

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

#### Restart

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

There are two ways to restart the daemon:

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

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

#### Security

By default, all of the <code>ckb\*</code> 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 <code>cmd</code> nodes or read from the <code>notify</code> nodes, you can specify the --gid=<group> option at start up. For instance, on most systems you could run <code>ckb-daemon --gid=1000</code> to make them accessible only by the system's primary user. <code>ckb-daemon must</code> still be run as root, regardless of which <code>gid</code> you specify. The <code>gid</code> 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 -qid, these options must be set at startup and cannot be changed while the daemon is running.

20 **DAEMON** 

# **Chapter 5**

# **Data Structure Index**

5.1	Data Structures	
Here	are the data structures with brief descriptions:	
do	wond uppomed	25

22 Data Structure Index

# **Chapter 6**

# File Index

# 6.1 File List

Here is a list of all files with brief descriptions:

	9
	4
	3
	6
- ,	5
	9
src/ckb-daemon/device_vtable.c	2
	4
src/ckb-daemon/devnode.h	6
	4
	9
	4
	5
src/ckb-daemon/firmware.h	0
src/ckb-daemon/includes.h	2
src/ckb-daemon/input.c	6
src/ckb-daemon/input.h	
src/ckb-daemon/input_linux.c	
src/ckb-daemon/input_mac.c	
src/ckb-daemon/input_mac_mouse.c	
src/ckb-daemon/keymap.c	
src/ckb-daemon/keymap.h	
src/ckb-daemon/keymap_mac.h	
src/ckb-daemon/led.c	
src/ckb-daemon/led.h	
src/ckb-daemon/led_keyboard.c	
src/ckb-daemon/led_mouse.c	
src/ckb-daemon/main.c	
src/ckb-daemon/notify.c	
src/ckb-daemon/notify.h	5
src/ckb-daemon/os.h	9
src/ckb-daemon/profile.c	!1
src/ckb-daemon/profile.h	
src/ckb-daemon/profile_keyboard.c	
src/ckb-daemon/profile_mouse.c	5
src/ckb-daemon/structures.h	
src/ckb-daemon/usb.c	6
src/ckb-daemon/usb.h	7

src/ckb-daemon/usb_	_linux.c				 														3	12
src/ckb-daemon/usb	mac.c					 											 		3	29

# **Chapter 7**

# **Data Structure Documentation**

7.1 devcmd.\_\_unnamed\_\_ Struct Reference

Collaboration diagram for devcmd.\_\_unnamed\_\_:

devcmd.\_\_unnamed\_\_

+ hwload
+ hwsave
+ fwupdate
+ pollrate
+ active
+ idle
+ erase
+ eraseprofile
+ name
+ profilename
and 26 more...

#### **Data Fields**

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

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

# 7.1.1 Detailed Description

Definition at line 78 of file command.h.

# 7.1.2 Field Documentation

7.1.2.1

7.1.2.2

7.1.2.3

7.1.2.4

7.1.2.5

7.1.2.6

7.1.2.7

7.1.2.8

7.1.2.9

7.1.2.10

7.1.2.11

7.1.2.12		
7.1.2.13		
7.1.2.14		
7.1.2.15		
7.1.2.16		
7.1.2.17		
7.1.2.18		
7.1.2.19		
7.1.2.20		
7.1.2.21		
7.1.2.22		
7.1.2.23		
7.1.2.24		
7.1.2.25		
7.1.2.26		
7.1.2.27		
7.1.2.28		
7.1.2.29		
7.1.2.30		
7.1.2.31		
7.1.2.32		
7.1.2.33		
7.1.2.34		
7.1.2.35		
7.1.2.36		

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



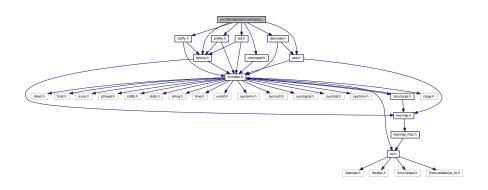
# **Chapter 8**

# **File Documentation**

- 8.1 BUILD.md File Reference
- 8.2 DAEMON.md File Reference
- 8.3 README.md File Reference
- 8.4 ROADMAP.md File Reference
- 8.5 src/ckb-daemon/command.c File Reference

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

Include dependency graph for command.c:



# **Macros**

• #define TRY\_WITH\_RESET(action)

#### **Functions**

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

#### **Variables**

static const char \*const cmd strings [(CMD LAST-CMD FIRST+2)-1]

#### 8.5.1 Macro Definition Documentation

```
8.5.1.1 #define TRY_WITH_RESET( action )
```

#### Value:

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

Definition at line 58 of file command.c.

Referenced by readcmd().

#### 8.5.2 Function Documentation

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

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

Definition at line 67 of file command.c.

References ACCEL, ACTIVE, usbdevice::active, BIND, CMD\_COUNT, CMD\_FIRST, cmd\_strings, usbprofile::currentmode, DELAY, usbdevice::delay, DITHER, usbdevice::dither, devcmd::do\_cmd, devcmd::do\_io, devcmd::do\_macro, DPI, DPISEL, ERASE, ERASEPROFILE, FEAT\_ANSI, FEAT\_BIND, FEAT\_ISO, FEAT\_LMASK, FEAT\_MOUSEACCEL, FEAT\_NOTIFY, usbdevice::features, lighting::forceupdate, FPS, FWUPDATE, GET, HAS\_FEATURES, HWLOAD, HWSAVE, IAUTO, ID, IDLE, INDEX\_OF, INOTIFY, IOFF, ION, IS\_FULLRANGE, IS\_MOUSE\_DEV, keymap, LAYOUT, LIFT, usbmode::light, MACRO, mknotifynode(), MODE, usbprofile::mode, MODE\_COUNT, N\_KEYS\_EXTENDED, NAME, NEEDS\_FW\_UPDATE, NONE, NOTIFY, NOTIFYOFF, NOTIFYON, OUTFIFO\_MAX, POLLRATE, usbdevice::profile, PROFILEID, PROFILENAME, REBIND, RESTART, RGB, rmnotifynode(), SCROLL\_ACCELERATED, SCROLL\_MAX, SCROLL\_MIN, SCROLLSPEED, SNAP, SWITCH, TRY\_WITH\_RESET, UNBIND, usbdevice::usbdelay, and usbdevice::vtable.

Referenced by devmain().

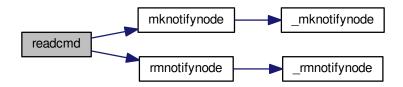
```
67
68
        char* word = malloc(strlen(line) + 1);
69
        int wordlen;
        const char* newline = 0;
70
        const devcmd* vt = kb->vtable;
71
        usbprofile* profile = kb->profile;
73
        usbmode* mode = 0;
74
        int notifynumber = 0;
        // Read words from the input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

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

```
164
                     newscroll = SCROLL_ACCELERATED;
                 if(newscroll > SCROLL_MAX)
   newscroll = SCROLL_MAX;
165
166
                 kb->scroll_rate = newscroll;
167
168
                 continue;
169
             }
170 #endif
171
             case MODE: {
172
                // Select a mode number (1 - 6)
173
                 int newmode;
                 if(sscanf(word, "%u", &newmode) == 1 && newmode > 0 && newmode <=</pre>
174
      MODE COUNT)
175
                     mode = profile->mode + newmode - 1;
176
                 continue;
177
             case FPS: {
178
                 // USB command delay (2 - 10ms)
179
180
                 uint framerate;
                 if(sscanf(word, "%u", &framerate) == 1 && framerate > 0) {
181
182
                      // Not all devices require the same number of messages per frame; select delay
       appropriately
183
                      uint per_frame = IS_MOUSE_DEV(kb) ? 2 : IS_FULLRANGE(kb) ? 14 : 5;
                     uint delay = 1000 / framerate / per_frame; if (delay < 2)
184
185
186
                         delay = 2;
187
                      else if(delay > 10)
188
                         delay = 10;
189
                      kb->usbdelay = delay;
190
                 }
191
                 continue:
192
193
             case DITHER: {
194
                 // 0: No dither, 1: Ordered dither.
195
                 uint dither;
                 if(sscanf(word, "%u", &dither) == 1 && dither <= 1) {
   kb->dither = dither;
   profile->currentmode->light.forceupdate = 1;
196
197
198
199
                     mode->light.forceupdate = 1;
200
                 }
201
                 continue;
202
203
             case DELAY:
                 kb->delay = (!strcmp (word, "on")); // independendant from parameter to handle false
2.04
       commands like "delay off"
205
                 continue;
206
             case RESTART:
                char mybuffer[] = "no reason specified";
if (sscanf(line, " %[^\n]", word) == -1) {
207
208
                     word = mybuffer;
209
210
211
                 vt->do_cmd[command] (kb, mode, notifynumber, 0, word);
212
213
             }
214
215
             default::
216
217
218
             // If a keyboard is inactive, it must be activated before receiving any other commands
             if(!kb->active){
219
220
                 if(command == ACTIVE)
                      TRY_WITH_RESET(vt->active(kb, mode, notifynumber, 0, 0));
221
222
                 continue;
223
             // Specially handled commands only available when keyboard is ACTIVE
224
225
             switch(command) {
             case IDLE:
226
227
                 TRY_WITH_RESET(vt->idle(kb, mode, notifynumber, 0, 0));
228
                 continue;
229
             case SWITCH:
                 if(profile->currentmode != mode) {
230
231
                     profile->currentmode = mode;
232
                      // Set mode light for non-RGB K95
233
                      int index = INDEX_OF(mode, profile->mode);
234
                      vt->setmodeindex(kb, index);
235
                 }
236
                 continue;
237
             case HWLOAD: case HWSAVE:{
238
                 char delay = kb->usbdelay;
                 // Ensure delay of at least 10ms as the device can get overwhelmed otherwise if(delay < 10) \,
239
240
241
                     kb->usbdelay = 10;
242
                  // Try to load/save the hardware profile. Reset on failure, disconnect if reset fails.
243
                 TRY_WITH_RESET(vt->do_io[command](kb, mode, notifynumber, 1, 0));
244
                 // Re-send the current RGB state as it sometimes gets scrambled
245
                 TRY_WITH_RESET(vt->updatergb(kb, 1));
246
                 kb->usbdelay = delay;
247
                 continue:
```

```
248
249
             case FWUPDATE:
250
                  // FW update parses a whole word. Unlike hwload/hwsave, there's no try again on failure.
2.51
                  if(vt->fwupdate(kb, mode, notifynumber, 0, word)){
2.52
                      free (word);
253
                      return 1:
254
255
                  continue;
256
             case POLLRATE: {
2.57
                  uint rate;
                  if(sscanf(word, "%u", &rate) == 1 && (rate == 1 || rate == 2 || rate == 4 || rate == 8))
258
                      TRY_WITH_RESET(vt->pollrate(kb, mode, notifynumber, rate, 0));
259
260
                  continue;
261
262
             case ERASEPROFILE:
263
                  \ensuremath{//} Erase the current profile
                  vt->eraseprofile(kb, mode, notifynumber, 0, 0);
264
                  // Update profile/mode pointers profile = kb->profile;
265
266
267
                  mode = profile->currentmode;
                  continue;
268
      case ERASE: case NAME: case IOFF: case ION: case IAUTO: case
INOTIFY: case PROFILENAME: case ID: case PROFILEID: case
DPISEL: case LIFT: case SNAP:
269
270
                  // All of the above just parse the whole word
271
                  vt->do_cmd[command](kb, mode, notifynumber, 0, word);
272
                  continue;
273
             case RGB: {
                  // RGB command has a special response for a single hex constant
274
                  int r, g, b;
if(sscanf(word, "%02x%02x%02x", &r, &g, &b) == 3){
275
276
                      // Set all keys
for(int i = 0; i < N_KEYS_EXTENDED; i++)</pre>
278
279
                           vt->rgb(kb, mode, notifynumber, i, word);
280
                       continue;
281
                  }
282
                  break:
283
284
             case MACRO:
285
                  if(!strcmp(word, "clear")){
286
                      // Macro has a special clear command
                      vt->macro(kb, mode, notifynumber, 0, 0);
287
288
                      continue:
289
                  }
290
                  break;
291
             default:;
292
             // For anything else, split the parameter at the colon
293
294
             int left = -1;
sscanf(word, "%*[^:]%n", &left);
295
296
             if(left <= 0)</pre>
297
                  continue;
298
             const char* right = word + left;
299
             if(right[0] == ':')
300
                  right++:
301
             // Macros and DPI have a separate left-side handler
             if (command == MACRO || command == DPI) {
302
303
                  word[left] = 0;
304
                  vt->do_macro[command](kb, mode, notifynumber, word, right);
305
                  continue;
306
             // Scan the left side for key names and run the requested command int position = 0, field = 0;
307
308
             char keyname[11];
309
310
             while (position < left && sscanf(word + position, "%10[^:,]%n", keyname, &field) == 1) {</pre>
311
                  int keycode;
                  if(!strcmp(keyname, "all")){
312
                      // Set all keys
for(int i = 0; i < N_KEYS_EXTENDED; i++)</pre>
313
314
                           vt->do_cmd[command](kb, mode, notifynumber, i, right);
315
316
                  } else if((sscanf(keyname, "#%d", &keycode) && keycode >= 0 && keycode <
      N_KEYS_EXTENDED)
317
                             || (sscanf(keyname, "\#x%x", &keycode) && keycode >= 0 && keycode <
      N KEYS EXTENDED)){
                       // Set a key numerically
318
319
                       vt->do_cmd[command](kb, mode, notifynumber, keycode, right);
320
                  } else {
                      // Find this key in the keymap
for(unsigned i = 0; i < N_KEYS_EXTENDED; i++){</pre>
321
322
                           if(keymap[i].name && !strcmp(keyname, keymap[i].name)){
323
324
                               vt->do_cmd[command](kb, mode, notifynumber, i, right);
325
                                break;
326
327
328
                  if(word[position += field] == ',')
329
330
                      position++;
```

Here is the call graph for this function:



Here is the caller graph for this function:



#### 8.5.3 Variable Documentation

**8.5.3.1** const char\* const cmd\_strings[(CMD\_LAST-CMD\_FIRST+2)-1] [static]

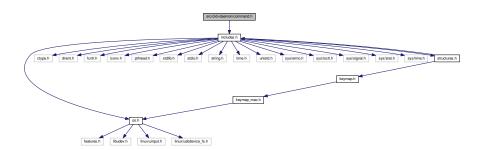
Definition at line 9 of file command.c.

Referenced by readcmd().

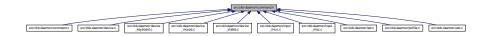
# 8.6 src/ckb-daemon/command.h File Reference

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

Include dependency graph for command.h:



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



#### **Data Structures**

- · union devcmd
- struct devcmd. unnamed

#### **Macros**

- #define CMD\_COUNT (CMD\_LAST CMD\_FIRST + 2)
- #define CMD\_DEV\_COUNT (CMD\_LAST CMD\_VT\_FIRST + 1)

#### **Typedefs**

- typedef void(\* cmdhandler )(usbdevice \*kb, usbmode \*modeidx, int notifyidx, int keyindex, const char \*parameter)
- typedef int(\* cmdhandler\_io )(usbdevice \*kb, usbmode \*modeidx, int notifyidx, int keyindex, const char \*parameter)
- typedef void(\* cmdhandler\_mac )(usbdevice \*kb, usbmode \*modeidx, int notifyidx, const char \*keys, const char \*assignment)
- typedef union devcmd devcmd

#### **Enumerations**

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

#### **Functions**

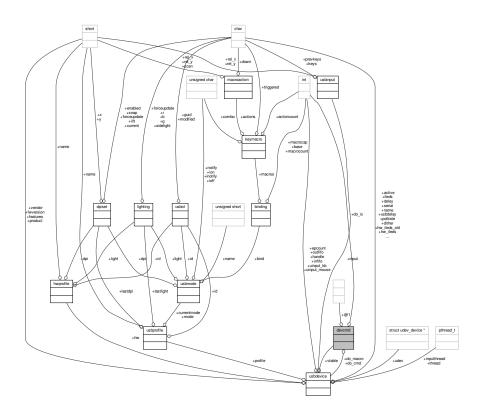
• int readcmd (usbdevice \*kb, const char \*line)

#### 8.6.1 Data Structure Documentation

# 8.6.1.1 union devcmd

Definition at line 73 of file command.h.

# Collaboration diagram for devcmd:



#### **Data Fields**

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

# 8.6.2 Macro Definition Documentation

# 8.6.2.1 #define CMD\_COUNT (CMD\_LAST - CMD\_FIRST + 2)

Definition at line 65 of file command.h.

Referenced by readcmd().

# $8.6.2.2 \quad \hbox{\#define CMD\_DEV\_COUNT (CMD\_LAST-CMD\_VT\_FIRST+1)}$

Definition at line 66 of file command.h.

# 8.6.3 Typedef Documentation

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

Definition at line 70 of file command.h.

8.6.3.2 typedef int(\* cmdhandler\_io)(usbdevice \*kb, usbmode \*modeidx, int notifyidx, int keyindex, const char \*parameter)

Definition at line 71 of file command.h.

8.6.3.3 typedef void(\* cmdhandler\_mac)(usbdevice \*kb, usbmode \*modeidx, int notifyidx, const char \*keys, const char \*assignment)

Definition at line 72 of file command.h.

- 8.6.3.4 typedef union devcmd devcmd
- 8.6.4 Enumeration Type Documentation
- 8.6.4.1 enum cmd

#### Enumerator

**NONE** 

**DELAY** 

CMD\_FIRST

MODE

**SWITCH** 

**LAYOUT** 

**ACCEL** 

**SCROLLSPEED** 

**NOTIFYON** 

**NOTIFYOFF** 

**FPS** 

**DITHER** 

**HWLOAD** 

CMD\_VT\_FIRST

**HWSAVE** 

**FWUPDATE** 

**POLLRATE** 

**ACTIVE** 

IDLE

**ERASE** 

**ERASEPROFILE** 

NAME

**PROFILENAME** 

ID

**PROFILEID** 

RGB

**IOFF** 

ION

IAUTO

**BIND** 

**UNBIND** 

REBIND

**MACRO** 

DPI

DPISEL

LIFT

**SNAP** 

**NOTIFY** 

**INOTIFY** 

**GET** 

RESTART

CMD\_LAST

Definition at line 7 of file command.h.

```
7
      // Special - handled by readcmd, no device functions NONE = -11,
8
                   = -11,
= -10,
= -9,
       DELAY
10
                               CMD_FIRST = DELAY,
11
        MODE
       SWITCH
                    = -8,
12
       LAYOUT
                    = -7,
13
        ACCEL
                     = -6,
14
        SCROLLSPEED = -5,
15
                    = -4,
= -3,
16
        NOTIFYON
17
       NOTIFYOFF
                     = -2,
        FPS
18
19
       DITHER
                    = -1,
20
21
        // Hardware data
22
        HWLOAD
                    = 0,
                             CMD\_VT\_FIRST = 0,
23
        HWSAVE,
       FWUPDATE,
2.4
25
        POLLRATE,
26
27
        // Software control on/off
28
29
        IDLE,
30
        // Profile/mode metadata
31
        ERASE,
32
33
        ERASEPROFILE,
34
        NAME,
35
        PROFILENAME,
       ID,
PROFILEID,
36
37
38
        // LED control
39
40
        RGB,
41
        IOFF,
42
        ION,
        IAUTO,
43
44
        // Key binding control
45
        BIND,
46
47
        UNBIND,
48
        REBIND,
       MACRO,
49
50
        // DPI control
51
       DPI,
DPISEL,
53
       SNAP,
55
56
        // Notifications and output
57
58
        NOTIFY,
        INOTIFY,
```

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

#### 8.6.5 Function Documentation

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

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

Definition at line 67 of file command.c.

References ACCEL, ACTIVE, usbdevice::active, BIND, CMD\_COUNT, CMD\_FIRST, cmd\_strings, usbprofile::currentmode, DELAY, usbdevice::delay, DITHER, usbdevice::dither, devcmd::do\_cmd, devcmd::do\_io, devcmd::do\_macro, DPI, DPISEL, ERASE, ERASEPROFILE, FEAT\_ANSI, FEAT\_BIND, FEAT\_ISO, FEAT\_LMASK, FEAT\_MOUSEACCEL, FEAT\_NOTIFY, usbdevice::features, lighting::forceupdate, FPS, FWUPDATE, GET, HAS\_FEATURES, HWLOAD, HWSAVE, IAUTO, ID, IDLE, INDEX\_OF, INOTIFY, IOFF, ION, IS\_FULLRANGE, IS\_MOUSE\_DEV, keymap, LAYOUT, LIFT, usbmode::light, MACRO, mknotifynode(), MODE, usbprofile::mode, MODE\_COUNT, N\_KEYS\_EXTENDED, NAME, NEEDS\_FW\_UPDATE, NONE, NOTIFY, NOTIFYOFF, NOTIFYON, OUTFIFO\_MAX, POLLRATE, usbdevice::profile, PROFILEID, PROFILENAME, REBIND, RESTART, RGB, rmnotifynode(), SCROLL\_ACCELERATED, SCROLL\_MAX, SCROLL\_MIN, SCROLLSPEED, SNAP, SWITCH, TRY\_WITH\_RESET, UNBIND, usbdevice::usbdelay, and usbdevice::vtable.

Referenced by devmain().

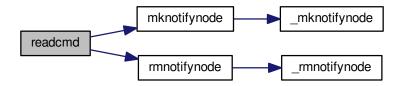
```
67
       char* word = malloc(strlen(line) + 1);
68
69
       int wordlen;
       const char* newline = 0;
70
       const devcmd* vt = kb->vtable;
       usbprofile* profile = kb->profile;
73
       usbmode* mode = 0;
       int notifynumber = 0;
74
75
       // Read words from the input
      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

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

```
193
            case DITHER: {
194
                // 0: No dither, 1: Ordered dither.
195
                 uint dither;
                 if(sscanf(word, "%u", &dither) == 1 && dither <= 1) {</pre>
196
                     kb->dither = dither;
profile->currentmode->light.forceupdate = 1;
197
198
199
                     mode->light.forceupdate = 1;
200
201
                 continue;
202
             case DELAY:
203
                kb->delay = (!strcmp (word, "on")); // independendant from parameter to handle false
204
       commands like "delay off"
205
                continue;
206
             case RESTART: {
                char mybuffer[] = "no reason specified";
if (sscanf(line, " %[^\n]", word) == -1) {
207
208
                     word = mybuffer;
209
210
211
                 vt->do_cmd[command](kb, mode, notifynumber, 0, word);
212
213
            }
214
            default:;
215
216
217
218
             // If a keyboard is inactive, it must be activated before receiving any other commands
219
             if(!kb->active){
220
                 if(command == ACTIVE)
221
                     TRY_WITH_RESET(vt->active(kb, mode, notifynumber, 0, 0));
222
                 continue:
223
224
            ^{\prime\prime} // Specially handled commands only available when keyboard is ACTIVE
225
             switch(command) {
            case IDLE:
226
                TRY_WITH_RESET(vt->idle(kb, mode, notifynumber, 0, 0));
227
228
                 continue;
             case SWITCH:
230
                if(profile->currentmode != mode) {
231
                    profile->currentmode = mode;
232
                      // Set mode light for non-RGB K95
                     int index = INDEX_OF(mode, profile->mode);
233
234
                     vt->setmodeindex(kb, index);
235
                 }
            continue;
case HWLOAD: case HWSAVE:{
236
237
238
                char delay = kb->usbdelay;
                 // Ensure delay of at least 10ms as the device can get overwhelmed otherwise if(delay < 10) \,
239
240
241
                     kb->usbdelav = 10;
242
                 // Try to load/save the hardware profile. Reset on failure, disconnect if reset fails.
243
                 TRY_WITH_RESET(vt->do_io[command](kb, mode, notifynumber, 1, 0));
244
                 \ensuremath{//} Re-send the current RGB state as it sometimes gets scrambled
245
                 TRY_WITH_RESET(vt->updatergb(kb, 1));
246
                 kb->usbdelay = delay;
247
                 continue;
248
249
            case FWUPDATE:
250
                 // FW update parses a whole word. Unlike hwload/hwsave, there's no try again on failure.
251
                 if(vt->fwupdate(kb, mode, notifynumber, 0, word)){
2.52
                     free (word);
253
                     return 1;
254
                 }
255
                 continue;
256
            case POLLRATE: {
257
                 uint rate;
                 if(sscanf(word, "%u", &rate) == 1 && (rate == 1 || rate == 2 || rate == 4 || rate == 8))
2.58
                     TRY_WITH_RESET(vt->pollrate(kb, mode, notifynumber, rate, 0));
259
260
                 continue:
261
262
            case ERASEPROFILE:
263
                 // Erase the current profile
2.64
                 vt->eraseprofile(kb, mode, notifynumber, 0, 0);
                 // Update profile/mode pointers
profile = kb->profile;
265
266
267
                 mode = profile->currentmode;
                 continue;
268
269
             case ERASE: case NAME: case IOFF: case ION: case IAUTO: case
      INOTIFY: case PROFILENAME: case ID: case PROFILEID: case
      DPISEL: case LIFT: case SNAP:
270
                // All of the above just parse the whole word
271
                 vt->do_cmd[command](kb, mode, notifynumber, 0, word);
272
273
             case RGB: {
274
                 \ensuremath{//}\xspace RGB command has a special response for a single hex constant
2.75
                 int r, g, b;
276
                 if(sscanf(word, "%02x%02x%02x", &r, &g, &b) == 3){
```

```
// Set all keys
278
                       for(int i = 0; i < N_KEYS_EXTENDED; i++)</pre>
279
                           vt->rgb(kb, mode, notifynumber, i, word);
                       continue;
280
2.81
282
                  break;
283
284
             case MACRO:
285
                  if(!strcmp(word, "clear")){
286
                      // Macro has a special clear command
                      vt->macro(kb, mode, notifynumber, 0, 0);
287
288
                      continue:
289
                  break;
290
291
             default:;
292
             // For anything else, split the parameter at the colon
293
294
             int left = -1;
sscanf(word, "%*[^:]%n", &left);
295
296
             if(left <= 0)
297
                 continue;
298
             const char* right = word + left;
             if(right[0] == ':')
299
300
                  right++;
301
             // Macros and DPI have a separate left-side handler
             if(command == MACRO || command == DPI) {
302
303
                  word[left] = 0;
304
                  vt->do_macro[command](kb, mode, notifynumber, word, right);
305
                  continue;
306
             ,// Scan the left side for key names and run the requested command int position = 0, field = 0;
307
308
309
             char keyname[11];
310
             while (position < left && sscanf(word + position, "%10[^:,]%n", keyname, &field) == 1){</pre>
311
                  int keycode;
                  if(!strcmp(keyname, "all")){
312
                      // Set all keys
for(int i = 0; i < N_KEYS_EXTENDED; i++)</pre>
313
314
                  vt->do_cmd[command](kb, mode, notifynumber, i, right);
} else if((sscanf(keyname, "#%d", &keycode) && keycode >= 0 && keycode <
315
316
      N_KEYS_EXTENDED)
317
                             || (sscanf(keyname, "#x%x", &keycode) && keycode >= 0 && keycode <
      N_KEYS_EXTENDED)) {
318
                      // Set a key numerically
319
                       vt->do_cmd[command](kb, mode, notifynumber, keycode, right);
320
                  } else {
                      // Find this key in the keymap
for(unsigned i = 0; i < N_KEYS_EXTENDED; i++){</pre>
321
322
                           if(keymap[i].name && !strcmp(keyname, keymap[i].name)){
323
324
                               vt->do_cmd[command](kb, mode, notifynumber, i, right);
325
                               break;
326
327
                       }
328
                  if(word[position += field] == ',')
329
330
                      position++;
331
             }
332
333
         // Finish up
334
         if (!NEEDS_FW_UPDATE(kb)) {
335
             TRY_WITH_RESET(vt->updatergb(kb, 0));
336
337
             TRY_WITH_RESET(vt->updatedpi(kb, 0));
338
339
         free (word);
340
         return 0;
341 }
```

Here is the call graph for this function:



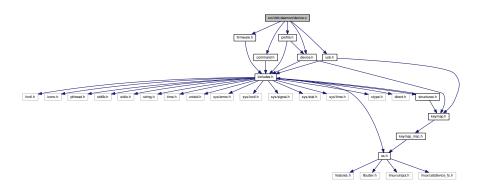
Here is the caller graph for this function:



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

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

Include dependency graph for device.c:



# **Functions**

- int <u>\_start\_dev</u> (usbdevice \*kb, int makeactive)
- int start\_dev (usbdevice \*kb, int makeactive)

#### **Variables**

- int hwload\_mode = 1
- usbdevice keyboard [9]

- pthread\_mutex\_t devlistmutex = PTHREAD\_MUTEX\_INITIALIZER
- pthread\_mutex\_t devmutex [9] = { [0 ... 9 -1] = PTHREAD\_MUTEX\_INITIALIZER }
- pthread\_mutex\_t inputmutex [9] = { [0 ... 9 -1] = PTHREAD\_MUTEX\_INITIALIZER }

#### 8.7.1 Function Documentation

#### 8.7.1.1 int start dev ( usbdevice \* kb, int makeactive )

Definition at line 15 of file device.c.

References usbdevice::active, ckb\_info, ckb\_warn, FEAT\_ADJRATE, FEAT\_FWUPDATE, FEAT\_FWVERSION, FEAT\_HWLOAD, FEAT\_POLLRATE, FEAT\_RGB, usbdevice::features, usbdevice::fwversion, getfwversion(), HAS\_FEATURES, usbdevice::hw, hwload\_mode, hwloadprofile, NEEDS\_FW\_UPDATE, usbdevice::pollrate, and setactive.

Referenced by start dev().

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

Here is the call graph for this function:



Here is the caller graph for this function:

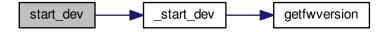


### 8.7.1.2 int start\_dev ( usbdevice \* kb, int makeactive )

Definition at line 55 of file device.c.

References \_start\_dev(), USB\_DELAY\_DEFAULT, and usbdevice::usbdelay.

Here is the call graph for this function:



#### 8.7.2 Variable Documentation

#### 8.7.2.1 pthread\_mutex\_t devlistmutex = PTHREAD\_MUTEX\_INITIALIZER

Definition at line 11 of file device.c.

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

Definition at line 12 of file device.c.

Referenced by \_updateconnected(), quitWithLock(), and usb\_rm\_device().

# 8.7.2.3 int hwload\_mode = 1

Definition at line 7 of file device.c.

Referenced by \_start\_dev(), \_usbrecv(), \_usbsend(), and main().

#### 8.7.2.4 pthread\_mutex\_t inputmutex[9] = { [0 ... 9 -1] = PTHREAD\_MUTEX\_INITIALIZER }

Definition at line 13 of file device.c.

#### 8.7.2.5 usbdevice keyboard[9]

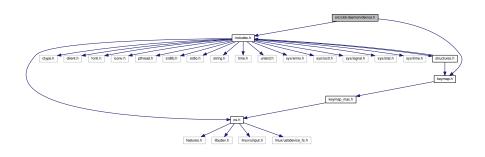
Definition at line 10 of file device.c.

Referenced by \_mkdevpath(), \_mknotifynode(), \_rmnotifynode(), \_setupusb(), \_updateconnected(), closeusb(), main(), mkfwnode(), os\_closeusb(), os\_inputmain(), os\_inputopen(), os\_setupusb(), quitWithLock(), rmdevpath(), usb\_rm\_device(), and usbadd().

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

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

Include dependency graph for device.h:



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



#### **Macros**

- #define DEV MAX 9
- #define IS\_CONNECTED(kb) ((kb) && (kb)->handle && (kb)->uinput\_kb && (kb)->uinput\_mouse)
- #define dmutex(kb) (devmutex + INDEX\_OF(kb, keyboard))
- #define imutex(kb) (inputmutex + INDEX\_OF(kb, keyboard))
- #define setactive(kb, makeactive) ((makeactive) ? (kb)->vtable->active((kb), 0, 0, 0, 0) : (kb)->vtable->idle((kb), 0, 0, 0, 0))
- #define IN\_HID 0x80
- #define IN CORSAIR 0x40
- #define ACT\_LIGHT 1
- #define ACT\_NEXT 3
- #define ACT\_NEXT\_NOWRAP 5
- #define ACT\_LOCK 8
- #define ACT MR RING 9
- #define ACT\_M1 10
- #define ACT\_M2 11
- #define ACT\_M3 12

#### **Functions**

- int start\_dev (usbdevice \*kb, int makeactive)
- int start kb nrgb (usbdevice \*kb, int makeactive)
- int setactive\_kb (usbdevice \*kb, int active)
- int setactive mouse (usbdevice \*kb, int active)
- int cmd\_active\_kb (usbdevice \*kb, usbmode \*dummy1, int dummy2, int dummy3, const char \*dummy4)
- int cmd active mouse (usbdevice \*kb, usbmode \*dummy1, int dummy2, int dummy3, const char \*dummy4)
- int cmd\_idle\_kb (usbdevice \*kb, usbmode \*dummy1, int dummy2, int dummy3, const char \*dummy4)
- int cmd\_idle\_mouse (usbdevice \*kb, usbmode \*dummy1, int dummy2, int dummy3, const char \*dummy4)
- int cmd\_pollrate (usbdevice \*kb, usbmode \*dummy1, int dummy2, int rate, const char \*dummy3)
- void setmodeindex\_nrgb (usbdevice \*kb, int index)

#### **Variables**

- usbdevice keyboard [9]
- pthread\_mutex\_t devmutex [9]
- pthread\_mutex\_t inputmutex [9]

#### 8.8.1 Macro Definition Documentation

8.8.1.1 #define ACT\_LIGHT 1

Definition at line 56 of file device.h.

Referenced by setactive\_kb().

8.8.1.2 #define ACT\_LOCK 8

Definition at line 59 of file device.h.

Referenced by setactive\_kb().

8.8.1.3 #define ACT\_M1 10

Definition at line 61 of file device.h.

Referenced by setactive kb().

8.8.1.4 #define ACT\_M2 11

Definition at line 62 of file device.h.

Referenced by setactive kb().

8.8.1.5 #define ACT\_M3 12

Definition at line 63 of file device.h.

Referenced by setactive\_kb().

8.8.1.6 #define ACT\_MR\_RING 9

Definition at line 60 of file device.h.

Referenced by setactive\_kb().

8.8.1.7 #define ACT\_NEXT 3

Definition at line 57 of file device.h.

8.8.1.8 #define ACT\_NEXT\_NOWRAP 5

Definition at line 58 of file device.h.

8.8.1.9 #define DEV\_MAX 9

Definition at line 8 of file device.h.

Referenced by \_updateconnected(), quitWithLock(), usb\_rm\_device(), and usbadd().

8.8.1.10 #define dmutex( kb ) (devmutex + INDEX\_OF(kb, keyboard))

Definition at line 18 of file device.h.

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

8.8.1.11 #define imutex( kb ) (inputmutex + INDEX\_OF(kb, keyboard))

Definition at line 22 of file device.h.

Referenced by \_setupusb(), closeusb(), cmd\_bind(), cmd\_erase(), cmd\_eraseprofile(), cmd\_get(), cmd\_macro(), cmd\_notify(), cmd\_rebind(), cmd\_unbind(), os\_inputmain(), setactive\_kb(), setactive\_mouse(), and setupusb().

8.8.1.12 #define IN\_CORSAIR 0x40

Definition at line 53 of file device.h.

Referenced by setactive\_kb(), and setactive\_mouse().

8.8.1.13 #define IN\_HID 0x80

Definition at line 52 of file device.h.

Referenced by setactive\_kb(), and setactive\_mouse().

8.8.1.14 #define IS\_CONNECTED( kb ) ((kb) && (kb)->handle && (kb)->uinput\_kb && (kb)->uinput\_mouse)

Definition at line 12 of file device.h.

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

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

Definition at line 32 of file device.h.

Referenced by \_start\_dev(), and revertusb().

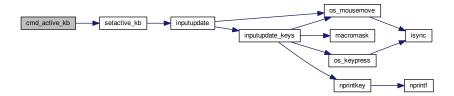
#### 8.8.2 Function Documentation

8.8.2.1 int cmd\_active\_kb ( usbdevice \* kb, usbmode \* dummy1, int dummy2, int dummy3, const char \* dummy4 )

Definition at line 112 of file device\_keyboard.c.

References setactive\_kb().

Here is the call graph for this function:



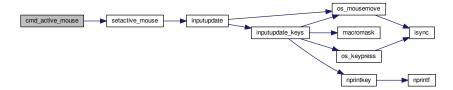
8.8.2.2 int cmd\_active\_mouse ( usbdevice \* kb, usbmode \* dummy1, int dummy2, int dummy3, const char \* dummy4 )

Definition at line 44 of file device\_mouse.c.

References setactive\_mouse().

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

Here is the call graph for this function:



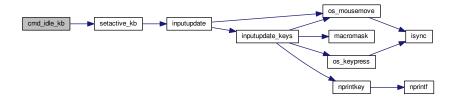
8.8.2.3 int cmd\_idle\_kb( usbdevice \* kb, usbmode \* dummy1, int dummy2, int dummy3, const char \* dummy4)

Definition at line 116 of file device\_keyboard.c.

References setactive\_kb().

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

Here is the call graph for this function:



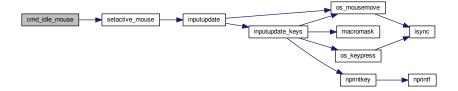
8.8.2.4 int cmd\_idle\_mouse ( usbdevice \* kb, usbmode \* dummy1, int dummy2, int dummy3, const char \* dummy4 )

Definition at line 48 of file device\_mouse.c.

References setactive\_mouse().

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

Here is the call graph for this function:



8.8.2.5 int cmd\_pollrate ( usbdevice \* kb, usbmode \* dummy1, int dummy2, int rate, const char \* dummy3 )

Definition at line 52 of file device\_mouse.c.

References MSG\_SIZE, usbdevice::pollrate, and usbsend.

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

8.8.2.6 int setactive\_kb ( usbdevice \* kb, int active )

Definition at line 18 of file device\_keyboard.c.

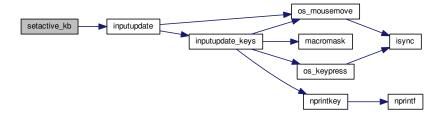
References ACT\_LIGHT, ACT\_LOCK, ACT\_M1, ACT\_M2, ACT\_M3, ACT\_MR\_RING, usbdevice::active, DELAY\_MEDIUM, lighting::forceupdate, imutex, IN\_CORSAIR, IN\_HID, usbdevice::input, inputupdate(), keymap, usbinput::keys, usbprofile::lastlight, MSG\_SIZE, N\_KEYS\_HW, NEEDS\_FW\_UPDATE, usbdevice::profile, usbsend, and usbdevice::vtable.

Referenced by cmd\_active\_kb(), and cmd\_idle\_kb().

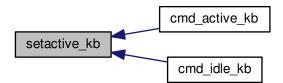
```
18
       if (NEEDS_FW_UPDATE(kb))
19
20
           return 0;
21
       pthread_mutex_lock(imutex(kb));
23
       kb->active = !!active;
       kb->profile->lastlight.forceupdate = 1;
24
       // Clear input
25
       memset(&kb->input.keys, 0, sizeof(kb->input.keys));
       inputupdate(kb);
28
       pthread_mutex_unlock(imutex(kb));
29
       uchar msg[3][MSG_SIZE] = {
30
           { 0x07, 0x04, 0 },
{ 0x07, 0x40, 0 },
                                                  // Disables or enables HW control for top row
31
32
                                                  // Selects key input
           { 0x07, 0x05, 2, 0, 0x03, 0x00 }
                                                 // Commits key input selection
34
35
       if(active){
36
           // Put the M-keys (K95) as well as the Brightness/Lock keys into software-controlled mode.
37
           msq[01[21 = 2;
38
           if(!usbsend(kb, msg[0], 1))
               return -1;
39
40
           DELAY_MEDIUM(kb);
           // Set input mode on the keys. They must be grouped into packets of 60 bytes (+ 4 bytes header)
41
           // Keys are referenced in byte pairs, with the first byte representing the key and the second byte
42
       representing the mode.
43
           for(int key = 0; key < N KEYS HW; ) {</pre>
               int pair;
                for(pair = 0; pair < 30 && key < N_KEYS_HW; pair++, key++){</pre>
45
46
                    // Select both standard and Corsair input. The standard input will be ignored except in
       BIOS mode.
47
                   uchar action = IN HID | IN CORSAIR:
48
                    // Additionally, make MR activate the MR ring (this is disabled for now, may be back later)
                    //if(keymap[key].name && !strcmp(keymap[key].name, "mr"))
49
                          action |= ACT_MR_RING;
                   msg[1][4 + pair * 2] = key;
msg[1][5 + pair * 2] = action;
51
52
53
               // Byte 2 = pair count (usually 30, less on final message)
54
               msg[1][2] = pair;
55
               if(!usbsend(kb, msg[1], 1))
57
                    return -1;
58
           // Commit new input settings
59
           if(!usbsend(kb, msg[2], 1))
60
               return -1;
61
           DELAY_MEDIUM(kb);
64
           // Set the M-keys back into hardware mode, restore hardware RGB profile. It has to be sent twice
       for some reason.
65
           msq[0][2] = 1;
66
           if(!usbsend(kb, msg[0], 1))
               return -1;
           DELAY_MEDIUM(kb);
68
69
           if(!usbsend(kb, msg[0], 1))
70
           DELAY_MEDIUM(kb);
72 #ifdef OS_LINUX
73
           // On OSX the default key mappings are fine. On Linux, the G keys will freeze the keyboard. Set the
       keyboard entirely to HID input.
74
           for(int key = 0; key < N_KEYS_HW; ){</pre>
7.5
               int pair;
               for(pair = 0; pair < 30 && key < N_KEYS_HW; pair++, key++){</pre>
76
                   uchar action = IN_HID;
                    // Enable hardware actions
78
79
                    if(keymap[key].name){
80
                        if(!strcmp(keymap[key].name, "mr"))
81
                            action = ACT_MR_RING;
82
                        else if(!strcmp(keymap[key].name, "m1"))
                            action = ACT M1;
83
                        else if(!strcmp(keymap[key].name, "m2"))
                            action = ACT_M2;
                        else if(!strcmp(keymap[key].name, "m3"))
87
                            action = ACT_M3;
                        else if(!strcmp(keymap[key].name, "light"))
     action = ACT_LIGHT;
88
89
                        else if(!strcmp(keymap[key].name, "lock"))
90
                            action = ACT_LOCK;
```

```
msg[1][4 + pair * 2] = key;
msg[1][5 + pair * 2] = action;
94
9.5
                   // Byte 2 = pair count (usually 30, less on final message)
96
                  msg[1][2] = pair;
if(!usbsend(kb, msg[1], 1))
98
99
100
              // Commit new input settings
if(!usbsend(kb, msg[2], 1))
101
102
103
                   return -1:
              DELAY_MEDIUM(kb);
104
105 #endif
106
107
          // Update indicator LEDs if the profile contains settings for them
108
         kb->vtable->updateindicators(kb, 0);
109
         return 0;
110 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



8.8.2.7 int setactive\_mouse ( usbdevice \* kb, int active )

Definition at line 9 of file device mouse.c.

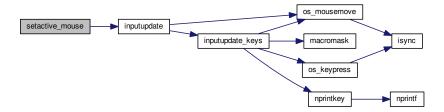
References usbdevice::active, lighting::forceupdate, imutex, IN\_CORSAIR, IN\_HID, usbdevice::input, inputupdate(), usbinput::keys, usbprofile::lastlight, MSG\_SIZE, NEEDS\_FW\_UPDATE, usbdevice::profile, and usbsend.

Referenced by cmd\_active\_mouse(), and cmd\_idle\_mouse().

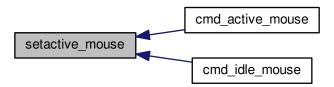
```
9
10    if (NEEDS_FW_UPDATE (kb))
11    return 0;
12    const int keycount = 20;
13    uchar msg[2][MSG_SIZE] = {
```

```
\{ 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
        if(active)
17
18
             // Put the mouse into SW mode
            msg[0][2] = 2;
19
20
            // Restore HW mode
21
22
            msg[0][2] = 1;
        pthread_mutex_lock(imutex(kb));
kb->active = !!active;
23
24
        kb->profile->lastlight.forceupdate = 1;
25
26
        // Clear input
27
        memset(&kb->input.keys, 0, sizeof(kb->input.keys));
28
        inputupdate(kb);
        pthread_mutex_unlock(imutex(kb));
29
        if(!usbsend(kb, msg[0], 1))
30
31
            return -1;
        if(active){
32
33
           // Set up key input
34
            if(!usbsend(kb, msg[1], 1))
            return -1;
for(int i = 0; i < keycount; i++) {
   msg[1][i * 2 + 4] = i + 1;
   msg[1][i * 2 + 5] = (i < 6 ? IN_HID : IN_CORSAIR);</pre>
3.5
36
39
40
41
        return 0;
42 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



8.8.2.8 void setmodeindex\_nrgb ( usbdevice \* kb, int index )

Definition at line 120 of file device\_keyboard.c.

References NK95\_M1, NK95\_M2, NK95\_M3, and nk95cmd.

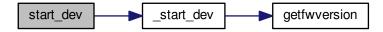
```
120
121
        switch(index % 3) {
122
        case 0:
            nk95cmd(kb, NK95_M1);
123
124
            break;
125
        case 1:
126
           nk95cmd(kb, NK95_M2);
127
            break;
128
        case 2:
129
           nk95cmd(kb, NK95_M3);
130
            break;
131
132 }
```

#### 8.8.2.9 int start\_dev ( usbdevice \* kb, int makeactive )

Definition at line 55 of file device.c.

References \_start\_dev(), USB\_DELAY\_DEFAULT, and usbdevice::usbdelay.

Here is the call graph for this function:



#### 8.8.2.10 int start\_kb\_nrgb ( usbdevice \* kb, int makeactive )

Definition at line 9 of file device keyboard.c.

References usbdevice::active, NK95\_HWOFF, nk95cmd, and usbdevice::pollrate.

## 8.8.3 Variable Documentation

### 8.8.3.1 pthread\_mutex\_t devmutex[9]

Definition at line 12 of file device.c.

Referenced by \_updateconnected(), quitWithLock(), and usb\_rm\_device().

#### 8.8.3.2 pthread\_mutex\_t inputmutex[9]

Definition at line 13 of file device.c.

#### 8.8.3.3 usbdevice keyboard[9]

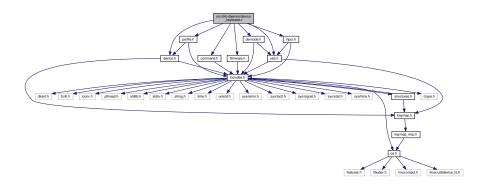
Definition at line 10 of file device.c.

Referenced by \_mkdevpath(), \_mknotifynode(), \_rmnotifynode(), \_setupusb(), \_updateconnected(), closeusb(), main(), mkfwnode(), os\_closeusb(), os\_inputmain(), os\_inputopen(), os\_setupusb(), quitWithLock(), rmdevpath(), usb\_rm\_device(), and usbadd().

## 8.9 src/ckb-daemon/device\_keyboard.c File Reference

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

Include dependency graph for device\_keyboard.c:



#### **Functions**

- int start kb nrgb (usbdevice \*kb, int makeactive)
- int setactive\_kb (usbdevice \*kb, int active)
- int cmd\_active\_kb (usbdevice \*kb, usbmode \*dummy1, int dummy2, int dummy3, const char \*dummy4)
- int cmd\_idle\_kb (usbdevice \*kb, usbmode \*dummy1, int dummy2, int dummy3, const char \*dummy4)
- void setmodeindex\_nrgb (usbdevice \*kb, int index)

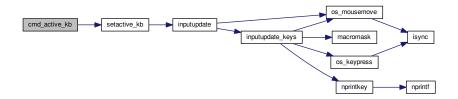
#### 8.9.1 Function Documentation

8.9.1.1 int cmd\_active\_kb ( usbdevice \* kb, usbmode \* dummy1, int dummy2, int dummy3, const char \* dummy4 )

Definition at line 112 of file device\_keyboard.c.

References setactive\_kb().

Here is the call graph for this function:



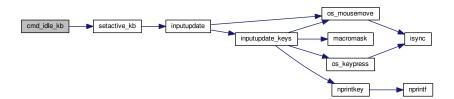
8.9.1.2 int cmd\_idle\_kb( usbdevice \* kb, usbmode \* dummy1, int dummy2, int dummy3, const char \* dummy4)

Definition at line 116 of file device\_keyboard.c.

References setactive\_kb().

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

Here is the call graph for this function:



8.9.1.3 int setactive\_kb ( usbdevice \* kb, int active )

Definition at line 18 of file device\_keyboard.c.

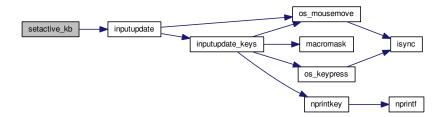
References ACT\_LIGHT, ACT\_LOCK, ACT\_M1, ACT\_M2, ACT\_M3, ACT\_MR\_RING, usbdevice::active, DELAY\_MEDIUM, lighting::forceupdate, imutex, IN\_CORSAIR, IN\_HID, usbdevice::input, inputupdate(), keymap, usbinput::keys, usbprofile::lastlight, MSG\_SIZE, N\_KEYS\_HW, NEEDS\_FW\_UPDATE, usbdevice::profile, usbsend, and usbdevice::vtable.

Referenced by cmd\_active\_kb(), and cmd\_idle\_kb().

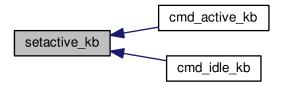
```
18
        if (NEEDS_FW_UPDATE(kb))
19
            return 0;
22
       pthread_mutex_lock(imutex(kb));
       kb->active = !!active;
kb->profile->lastlight.forceupdate = 1;
23
24
25
        // Clear input
26
       memset(&kb->input.keys, 0, sizeof(kb->input.keys));
        inputupdate(kb);
28
 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

```
33
            { 0x07, 0x05, 2, 0, 0x03, 0x00 } // Commits key input selection
34
35
        if(active){
36
            // Put the M-keys (K95) as well as the Brightness/Lock keys into software-controlled mode.
37
            msq[0][2] = 2;
            if(!usbsend(kb, msg[0], 1))
38
39
                 return -1;
            DELAY_MEDIUM(kb);
40
            // Set input mode on the keys. They must be grouped into packets of 60 bytes (+ 4 bytes header)
// Keys are referenced in byte pairs, with the first byte representing the key and the second byte
41
42
        representing the mode.
43
            for(int kev = 0; kev < N KEYS HW; ) {</pre>
                 int pair;
44
                 for(pair = 0; pair < 30 && key < N_KEYS_HW; pair++, key++){</pre>
45
46
                      // Select both standard and Corsair input. The standard input will be ignored except in
        BIOS mode.
47
                     uchar action = IN HID | IN CORSAIR;
                     // Additionally, make MR activate the MR ring (this is disabled for now, may be back later) //if(keymap[key].name && !strcmp(keymap[key].name, "mr"))
48
49
                            action |= ACT_MR_RING;
                     msg[1][4 + pair * 2] = key;
msg[1][5 + pair * 2] = action;
52
5.3
                 // Byte 2 = pair count (usually 30, less on final message)
54
                 msq[1][2] = pair;
55
                 if(!usbsend(kb, msg[1], 1))
56
57
                      return -1;
58
            // Commit new input settings
59
            if(!usbsend(kb, msg[2], 1))
60
                 return -1:
61
            DELAY_MEDIUM(kb);
62
       } else {
64
            // Set the M-keys back into hardware mode, restore hardware RGB profile. It has to be sent twice
        for some reason.
65
            msq[0][2] = 1;
            if(!usbsend(kb, msg[0], 1))
66
                return -1;
            DELAY_MEDIUM(kb);
69
            if(!usbsend(kb, msg[0], 1))
70
                 return -1:
            DELAY MEDIUM (kb):
71
   #ifdef OS_LINUX
72
73
            // On OSX the default key mappings are fine. On Linux, the G keys will freeze the keyboard. Set the
        keyboard entirely to HID input.
74
            for(int key = 0; key < N_KEYS_HW; ) {</pre>
75
                 int pair;
                 for(pair = 0; pair < 30 && key < N_KEYS_HW; pair++, key++){
    uchar action = IN_HID;</pre>
76
77
78
                     // Enable hardware actions
                      if (keymap[key].name) {
80
                          if(!strcmp(keymap[key].name, "mr"))
81
                              action = ACT_MR_RING;
82
                          else if(!strcmp(keymap[key].name, "m1"))
83
                              action = ACT M1;
                          else if(!strcmp(keymap[key].name, "m2"))
84
                              action = ACT_M2;
86
                          else if(!strcmp(keymap[key].name, "m3"))
87
                              action = ACT_M3;
88
                          else if(!strcmp(keymap[key].name, "light"))
89
                              action = ACT LIGHT;
90
                          else if(!strcmp(keymap[key].name, "lock"))
91
                              action = ACT_LOCK;
                     msg[1][4 + pair * 2] = key;
msg[1][5 + pair * 2] = action;
93
94
9.5
                 // Byte 2 = pair count (usually 30, less on final message)
96
97
                 msg[1][2] = pair;
                 if(!usbsend(kb, msg[1], 1))
99
                     return -1;
100
101
             // Commit new input settings
             if(!usbsend(kb, msg[2], 1))
102
103
             DELAY_MEDIUM(kb);
104
105 #endif
106
         ^{\prime} // Update indicator LEDs if the profile contains settings for them
107
108
         kb->vtable->updateindicators(kb, 0);
109
         return 0;
110 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



## 8.9.1.4 void setmodeindex\_nrgb ( usbdevice \* kb, int index )

Definition at line 120 of file device\_keyboard.c.

References NK95\_M1, NK95\_M2, NK95\_M3, and nk95cmd.

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

### 8.9.1.5 int start\_kb\_nrgb ( usbdevice \* kb, int makeactive )

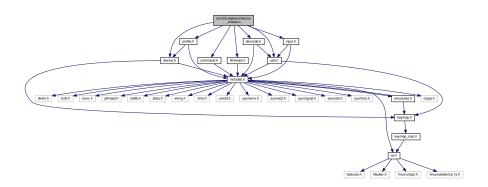
Definition at line 9 of file device\_keyboard.c.

References usbdevice::active, NK95\_HWOFF, nk95cmd, and usbdevice::pollrate.

## 8.10 src/ckb-daemon/device mouse.c File Reference

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

Include dependency graph for device\_mouse.c:



#### **Functions**

- int setactive\_mouse (usbdevice \*kb, int active)
- int cmd\_active\_mouse (usbdevice \*kb, usbmode \*dummy1, int dummy2, int dummy3, const char \*dummy4)
- int cmd\_idle\_mouse (usbdevice \*kb, usbmode \*dummy1, int dummy2, int dummy3, const char \*dummy4)
- int cmd\_pollrate (usbdevice \*kb, usbmode \*dummy1, int dummy2, int rate, const char \*dummy3)

#### 8.10.1 Function Documentation

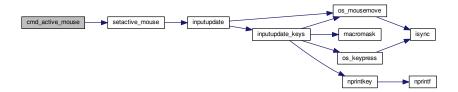
8.10.1.1 int cmd\_active\_mouse ( usbdevice \* kb, usbmode \* dummy1, int dummy2, int dummy3, const char \* dummy4 )

Definition at line 44 of file device\_mouse.c.

References setactive\_mouse().

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

Here is the call graph for this function:



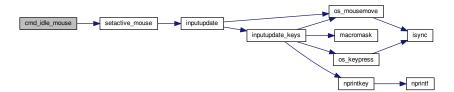
8.10.1.2 int cmd\_idle\_mouse ( usbdevice \* kb, usbmode \* dummy1, int dummy2, int dummy3, const char \* dummy4 )

Definition at line 48 of file device mouse.c.

References setactive\_mouse().

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

Here is the call graph for this function:



8.10.1.3 int cmd\_pollrate ( usbdevice \* kb, usbmode \* dummy1, int dummy2, int rate, const char \* dummy3 )

Definition at line 52 of file device\_mouse.c.

References MSG\_SIZE, usbdevice::pollrate, and usbsend.

8.10.1.4 int setactive\_mouse ( usbdevice \* kb, int active )

Definition at line 9 of file device mouse.c.

References usbdevice::active, lighting::forceupdate, imutex, IN\_CORSAIR, IN\_HID, usbdevice::input, inputupdate(), usbinput::keys, usbprofile::lastlight, MSG\_SIZE, NEEDS\_FW\_UPDATE, usbdevice::profile, and usbsend.

Referenced by cmd\_active\_mouse(), and cmd\_idle\_mouse().

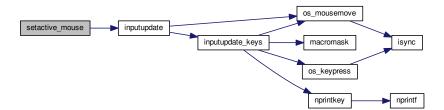
```
{
10
         if (NEEDS_FW_UPDATE(kb))
11
              return 0;
        const int keycount = 20;

uchar msg[2][MSG_SIZE] = {

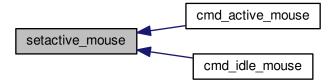
{ 0x07, 0x04, 0 },

{ 0x07, 0x40, keycount, 0 },
12
1.3
                                                             // Disables or enables HW control for DPI and Sniper button
14
                                                             // Select button input (simlilar to the packet sent to
15
         keyboards, but lacks a commit packet)
16
17
         if(active)
             // Put the mouse into SW mode
msg[0][2] = 2;
18
19
20
         else
21
              // Restore HW mode
22
              msg[0][2] = 1;
         pthread_mutex_lock(imutex(kb));
23
         kb->active = !!active;
kb->profile->lastlight.forceupdate = 1;
24
25
         // Clear input
26
         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))
              return -1;
for(int i = 0; i < keycount; i++) {
   msg[1][i * 2 + 4] = i + 1;
   msg[1][i * 2 + 5] = (i < 6 ? IN_HID : IN_CORSAIR);</pre>
35
36
37
38
39
40
41
         return 0;
42 }
```

Here is the call graph for this function:



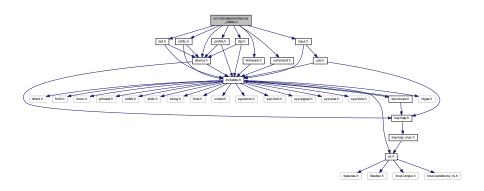
Here is the caller graph for this function:



## 8.11 src/ckb-daemon/device\_vtable.c File Reference

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

Include dependency graph for device\_vtable.c:



#### **Functions**

- static void cmd\_none (usbdevice \*kb, usbmode \*dummy1, int dummy2, int dummy3, const char \*dummy4)
- static int cmd\_io\_none (usbdevice \*kb, usbmode \*dummy1, int dummy2, int dummy3, const char \*dummy4)
- static void cmd\_macro\_none (usbdevice \*kb, usbmode \*dummy1, int dummy2, const char \*dummy3, const char \*dummy4)
- static int loadprofile\_none (usbdevice \*kb)
- static void int1\_void\_none (usbdevice \*kb, int dummy)
- static int int1\_int\_none (usbdevice \*kb, int dummy)

### **Variables**

- · const devcmd vtable\_keyboard
- · const devcmd vtable\_keyboard\_nonrgb
- · const devcmd vtable mouse

### 8.11.1 Function Documentation

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

Definition at line 13 of file device\_vtable.c.

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

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

Definition at line 16 of file device\_vtable.c.

```
16
17 }
```

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

Definition at line 11 of file device vtable.c.

```
11
12 }
```

8.11.1.4 static int int1\_int\_none ( usbdevice \* kb, int dummy ) [static]

Definition at line 23 of file device\_vtable.c.

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

8.11.1.5 static void int1\_void\_none( usbdevice \* kb, int dummy ) [static]

Definition at line 21 of file device\_vtable.c.

```
21 {
```

**8.11.1.6** static int loadprofile\_none ( usbdevice \* kb ) [static]

Definition at line 18 of file device\_vtable.c.

### 8.11.2 Variable Documentation

8.11.2.1 const devcmd vtable\_keyboard

Definition at line 28 of file device\_vtable.c.

Referenced by get\_vtable().

8.11.2.2 const devcmd vtable\_keyboard\_nonrgb

Definition at line 75 of file device\_vtable.c.

Referenced by get\_vtable().

#### 8.11.2.3 const devcmd vtable\_mouse

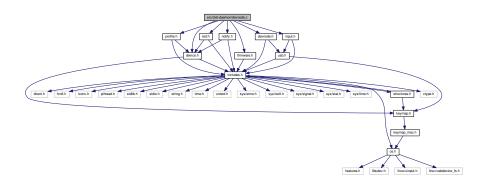
Definition at line 122 of file device\_vtable.c.

Referenced by get\_vtable().

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

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

Include dependency graph for devnode.c:



#### **Data Structures**

· struct \_readlines\_ctx

#### **Macros**

- #define S\_GID\_READ (gid >= 0 ? S\_CUSTOM\_R : S\_READ)
- #define MAX BUFFER (1024 \* 1024 1)

### **Functions**

- int rm\_recursive (const char \*path)
- void \_updateconnected ()
- void updateconnected ()
- int \_mknotifynode (usbdevice \*kb, int notify)
- int mknotifynode (usbdevice \*kb, int notify)
- int \_rmnotifynode (usbdevice \*kb, int notify)
- int rmnotifynode (usbdevice \*kb, int notify)
- static int \_mkdevpath (usbdevice \*kb)
- int mkdevpath (usbdevice \*kb)
- int rmdevpath (usbdevice \*kb)
- int mkfwnode (usbdevice \*kb)
- void readlines\_ctx\_init (readlines\_ctx \*ctx)

- void readlines\_ctx\_free (readlines\_ctx ctx)
- unsigned readlines (int fd, readlines\_ctx ctx, const char \*\*input)

## **Variables**

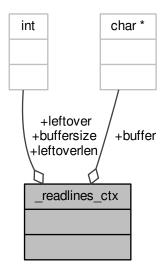
- const char \*const devpath = "/dev/input/ckb"
- long gid = -1

#### 8.12.1 Data Structure Documentation

8.12.1.1 struct \_readlines\_ctx

Definition at line 318 of file devnode.c.

Collaboration diagram for \_readlines\_ctx:



#### **Data Fields**

char *	buffer	
int	buffersize	
int	leftover	
int	leftoverlen	

### 8.12.2 Macro Definition Documentation

8.12.2.1 #define MAX\_BUFFER (1024 \* 1024 - 1)

Definition at line 317 of file devnode.c.

Referenced by readlines().

#### 8.12.2.2 #define S\_GID\_READ (gid >= 0 ? S\_CUSTOM\_R : S\_READ)

Definition at line 17 of file devnode.c.

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

#### 8.12.3 Function Documentation

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

Definition at line 119 of file devnode.c.

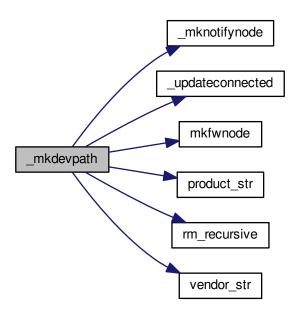
References \_mknotifynode(), \_updateconnected(), ckb\_err, ckb\_warn, devpath, FEAT\_ADJRATE, FEAT\_BIND, FEAT\_FWUPDATE, FEAT\_FWVERSION, FEAT\_MONOCHROME, FEAT\_NOTIFY, FEAT\_POLLRATE, FEAT\_RGB, gid, HAS\_FEATURES, INDEX\_OF, usbdevice::infifo, keyboard, mkfwnode(), usbdevice::name, usbdevice::product, product\_str(), rm\_recursive(), S\_CUSTOM, S\_GID\_READ, S\_READ, S\_READDIR, S\_READWRITE, usbdevice::serial, usbdevice::vendor, and vendor\_str().

Referenced by mkdevpath().

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

```
kb->infifo = 0;
178
                     return -1;
179
                if(gid >= 0)
180
                     fchown(kb->infifo - 1, 0, gid);
181
182
183
                // Create notification FIFO
184
                _mknotifynode(kb, 0);
185
                \ensuremath{//} Write the model and serial to files
186
                // white the model and serial to files
char mpath[sizeof(path) + 6], spath[sizeof(path) + 7];
snprintf(mpath, sizeof(mpath), "%s/model", path);
snprintf(spath, sizeof(spath), "%s/serial", path);
FILE* mfile = fopen(mpath, "w");
187
188
189
190
191
                if(mfile) {
                     fputs(kb->name, mfile);
fputc('\n', mfile);
fclose(mfile);
192
193
194
195
                     chmod(mpath, S_GID_READ);
196
                      if(gid >= 0)
197
                           chown (mpath, 0, gid);
198
                } else {
                     \label{linear_ckb_warn("Unable to create %s: %s/n", mpath, strerror(errno));}
199
200
                     remove (mpath);
201
                FILE* sfile = fopen(spath, "w");
203
                if(sfile){
204
                     fputs(kb->serial, sfile);
                     fputc('\n', sfile);
fclose(sfile);
205
206
207
                     chmod(spath, S_GID_READ);
208
                     if (gid >= 0)
209
                          chown(spath, 0, gid);
210
211
                     ckb_warn("Unable to create %s: %s\n", spath, strerror(errno));
212
                     remove(spath);
213
                // Write the keyboard's features
215
                char fpath[sizeof(path) + 9];
216
                snprintf(fpath, sizeof(fpath), "%s/features", path);
                FILE* ffile = fopen(fpath, "w");
217
                if(ffile){
218
                     fprintf(ffile, "%s %s", vendor_str(kb->vendor),
219
        product_str(kb->product));
220
                     if(HAS_FEATURES(kb, FEAT_MONOCHROME))
                     fputs(" monochrome", ffile);
if(HAS_FEATURES(kb, FEAT_RGB))
  fputs(" rgb", ffile);
if(HAS_FEATURES(kb, FEAT_POLLRATE))
  fputs(" pollrate", ffile);
221
222
223
224
225
                     if(HAS_FEATURES(kb, FEAT_ADJRATE))
226
227
                           fputs(" adjrate", ffile);
228
                     if(HAS_FEATURES(kb, FEAT_BIND))
                     fputs(" bind", ffile);
if(HAS_FEATURES(kb, FEAT_NOTIFY))
fputs(" notify", ffile);
if(HAS_FEATURES(kb, FEAT_FWVERSION))
229
230
231
                     fputs(" fwversion", ffile);
if(HAS_FEATURES(kb, FEAT_FWUPDATE))
233
234
                     fputs(" fwupdate", ffile);
fputc('\n', ffile);
fclose(ffile);
235
236
237
238
                     chmod(fpath, S_GID_READ);
239
                     if (gid >= 0)
240
                           chown(fpath, 0, gid);
241
                } else {
242
                     ckb_warn("Unable to create %s: %s\n", fpath, strerror(errno));
243
                     remove (fpath);
244
                ^{\prime} // Write firmware version and poll rate
245
246
                mkfwnode(kb);
2.47
248
           return 0;
249 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



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

Definition at line 70 of file devnode.c.

References ckb\_warn, devpath, gid, INDEX\_OF, keyboard, usbdevice::outfifo, OUTFIFO\_MAX, and S\_GID\_READ. Referenced by \_mkdevpath(), and mknotifynode().

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

Here is the caller graph for this function:



#### 8.12.3.3 int \_rmnotifynode ( usbdevice \* kb, int notify )

Definition at line 98 of file devnode.c.

References devpath, INDEX\_OF, keyboard, usbdevice::outfifo, and OUTFIFO\_MAX.

Referenced by rmdevpath(), and rmnotifynode().

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

Here is the caller graph for this function:



### 8.12.3.4 void \_updateconnected ( )

Definition at line 38 of file devnode.c.

References ckb\_warn, DEV\_MAX, devmutex, devpath, gid, IS\_CONNECTED, keyboard, and S\_GID\_READ.

Referenced by \_mkdevpath(), and updateconnected().

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

Here is the caller graph for this function:



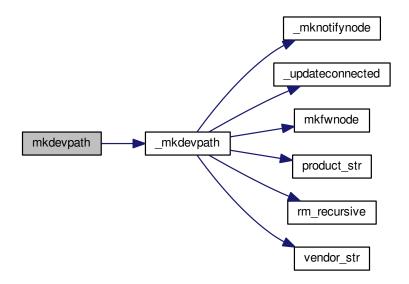
## 8.12.3.5 int mkdevpath ( usbdevice \* kb )

Definition at line 251 of file devnode.c.

References \_mkdevpath(), euid\_guard\_start, and euid\_guard\_stop.

Referenced by \_setupusb(), and main().

Here is the call graph for this function:



Here is the caller graph for this function:



#### 8.12.3.6 int mkfwnode ( usbdevice \*kb )

Definition at line 282 of file devnode.c.

References ckb\_warn, devpath, usbdevice::fwversion, gid, INDEX\_OF, keyboard, usbdevice::pollrate, and S\_GID\_-READ.

Referenced by \_mkdevpath(), and fwupdate().

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

Here is the caller graph for this function:



#### 8.12.3.7 int mknotifynode ( usbdevice \* kb, int notify )

Definition at line 91 of file devnode.c.

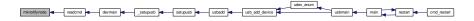
References \_mknotifynode(), euid\_guard\_start, and euid\_guard\_stop.

Referenced by readcmd().

Here is the call graph for this function:



Here is the caller graph for this function:



#### 8.12.3.8 unsigned readlines ( int fd, readlines\_ctx ctx, const char \*\* input )

Definition at line 336 of file devnode.c.

References \_readlines\_ctx::buffer, \_readlines\_ctx::buffersize, ckb\_warn, \_readlines\_ctx::leftover, \_readlines\_ctx::leftoverlen, and MAX\_BUFFER.

Referenced by devmain().

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

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

Here is the caller graph for this function:



### 8.12.3.9 void readlines\_ctx\_free ( readlines\_ctx ctx )

Definition at line 331 of file devnode.c.

References \_readlines\_ctx::buffer.

Referenced by devmain().

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

Here is the caller graph for this function:

#### 8.12.3.10 void readlines\_ctx\_init ( readlines\_ctx \* ctx )

Definition at line 324 of file devnode.c.

Referenced by devmain().

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

Here is the caller graph for this function:



#### 8.12.3.11 int rm\_recursive ( const char \* path )

Definition at line 19 of file devnode.c.

Referenced by \_mkdevpath(), and rmdevpath().

```
20
       DIR* dir = opendir(path);
21
       if(!dir)
2.2
            return remove (path);
23
       struct dirent* file;
24
       while((file = readdir(dir)))
25
26
            if(!strcmp(file->d_name, ".") || !strcmp(file->d_name, ".."))
27
            char path2[FILENAME_MAX];
2.8
            snprintf(path2, FILENAME_MAX, "%s/%s", path, file->d_name);
29
            int stat = rm_recursive(path2);
if(stat != 0)
30
31
32
                return stat;
33
34
       closedir (dir);
35
       return remove(path);
36 }
```

Here is the caller graph for this function:



#### 8.12.3.12 int rmdevpath ( usbdevice \*kb )

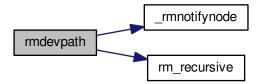
Definition at line 258 of file devnode.c.

 $References \_rmnotifynode(), \ ckb\_info, \ ckb\_warn, \ devpath, \ euid\_guard\_start, \ euid\_guard\_stop, \ INDEX\_OF, \ usbdevice::infifo, keyboard, OUTFIFO\_MAX, and rm\_recursive().$ 

Referenced by closeusb(), and quitWithLock().

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

Here is the call graph for this function:



Here is the caller graph for this function:



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

Definition at line 112 of file devnode.c.

References \_rmnotifynode(), euid\_guard\_start, and euid\_guard\_stop.

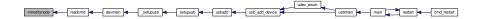
Referenced by readcmd().

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

Here is the call graph for this function:



Here is the caller graph for this function:



```
8.12.3.14 void updateconnected ( )
```

Definition at line 64 of file devnode.c.

References \_updateconnected(), euid\_guard\_start, and euid\_guard\_stop.

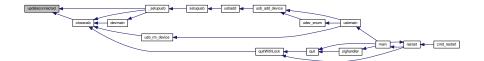
Referenced by \_setupusb(), and closeusb().

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

Here is the call graph for this function:



Here is the caller graph for this function:



## 8.12.4 Variable Documentation

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

Definition at line 11 of file devnode.c.

Referenced by \_mkdevpath(), \_mknotifynode(), \_rmnotifynode(), \_setupusb(), \_updateconnected(), closeusb(), main(), mkfwnode(), os inputmain(), os setupusb(), and rmdevpath().

```
8.12.4.2 long gid = -1
```

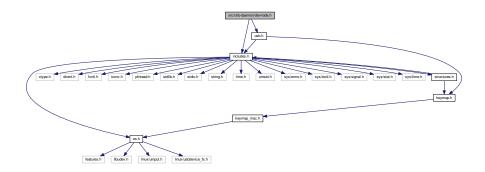
Definition at line 16 of file devnode.c.

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

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

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

Include dependency graph for devnode.h:



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



#### **Macros**

- #define S\_READDIR (S\_IRWXU | S\_IRGRP | S\_IROTH | S\_IXGRP | S\_IXOTH)
- #define S\_READ (S\_IRUSR | S\_IRGRP | S\_IROTH | S\_IWUSR)
- #define S\_READWRITE (S\_IRUSR | S\_IRGRP | S\_IROTH | S\_IWUSR | S\_IWGRP | S\_IWOTH)
- #define S\_CUSTOM (S\_IRUSR | S\_IWUSR | S\_IRGRP | S\_IWGRP)
- #define S\_CUSTOM\_R (S\_IRUSR | S\_IWUSR | S\_IRGRP)

### **Typedefs**

typedef struct \_readlines\_ctx \* readlines\_ctx

### **Functions**

- void updateconnected ()
- int mkdevpath (usbdevice \*kb)
- int rmdevpath (usbdevice \*kb)
- int mknotifynode (usbdevice \*kb, int notify)
- int rmnotifynode (usbdevice \*kb, int notify)
- int mkfwnode (usbdevice \*kb)
- void readlines\_ctx\_init (readlines\_ctx \*ctx)
- void readlines\_ctx\_free (readlines\_ctx ctx)
- unsigned readlines (int fd, readlines\_ctx ctx, const char \*\*input)

### **Variables**

- const char \*const devpath
- · long gid

### 8.13.1 Macro Definition Documentation

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

Definition at line 17 of file devnode.h.

Referenced by \_mkdevpath().

```
8.13.1.2 #define S_CUSTOM_R (S_IRUSR | S_IWUSR | S_IRGRP)
```

Definition at line 18 of file devnode.h.

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

Definition at line 15 of file devnode.h.

Referenced by \_mkdevpath().

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

Definition at line 14 of file devnode.h.

Referenced by \_mkdevpath().

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

Definition at line 16 of file devnode.h.

Referenced by \_mkdevpath().

### 8.13.2 Typedef Documentation

8.13.2.1 typedef struct \_readlines\_ctx\* readlines\_ctx

Definition at line 36 of file devnode.h.

### 8.13.3 Function Documentation

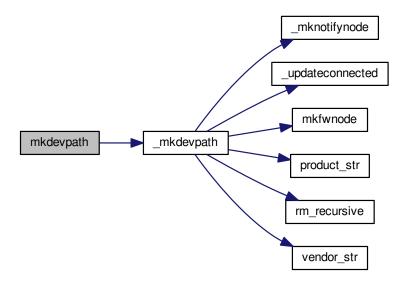
```
8.13.3.1 int mkdevpath ( usbdevice * kb )
```

Definition at line 251 of file devnode.c.

References \_mkdevpath(), euid\_guard\_start, and euid\_guard\_stop.

Referenced by \_setupusb(), and main().

Here is the call graph for this function:



Here is the caller graph for this function:



### 8.13.3.2 int mkfwnode ( usbdevice \* kb )

Definition at line 282 of file devnode.c.

References ckb\_warn, devpath, usbdevice::fwversion, gid, INDEX\_OF, keyboard, usbdevice::pollrate, and S\_GID\_-READ.

Referenced by \_mkdevpath(), and fwupdate().

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

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

Here is the caller graph for this function:



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

Definition at line 91 of file devnode.c.

References \_mknotifynode(), euid\_guard\_start, and euid\_guard\_stop.

Referenced by readcmd().

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

Here is the call graph for this function:



Here is the caller graph for this function:



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

Definition at line 336 of file devnode.c.

References \_readlines\_ctx::buffer, \_readlines\_ctx::buffersize, ckb\_warn, \_readlines\_ctx::leftover, \_readlines\_ctx::leftoverlen, and MAX\_BUFFER.

Referenced by devmain().

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

Here is the caller graph for this function:



8.13.3.5 void readlines\_ctx\_free ( readlines\_ctx ctx )

Definition at line 331 of file devnode.c.

References \_readlines\_ctx::buffer.

Referenced by devmain().

331

Here is the caller graph for this function:

```
readfires; ctx, free downain setupusb usbadd usb, add, device usb main restart cmd, restart usb main restart
```

8.13.3.6 void readlines\_ctx\_init ( readlines\_ctx \* ctx )

Definition at line 324 of file devnode.c.

Referenced by devmain().

Here is the caller graph for this function:

#### 8.13.3.7 int rmdevpath ( usbdevice \* kb )

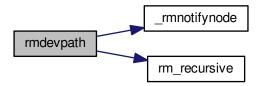
Definition at line 258 of file devnode.c.

References \_rmnotifynode(), ckb\_info, ckb\_warn, devpath, euid\_guard\_start, euid\_guard\_stop, INDEX\_OF, usbdevice::infifo, keyboard, OUTFIFO\_MAX, and rm\_recursive().

Referenced by closeusb(), and quitWithLock().

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

Here is the call graph for this function:



Here is the caller graph for this function:



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

Definition at line 112 of file devnode.c.

References \_rmnotifynode(), euid\_guard\_start, and euid\_guard\_stop.

Referenced by readcmd().

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

Here is the call graph for this function:



Here is the caller graph for this function:



```
8.13.3.9 void updateconnected ( )
```

Definition at line 64 of file devnode.c.

References \_updateconnected(), euid\_guard\_start, and euid\_guard\_stop.

Referenced by setupusb(), and closeusb().

Here is the call graph for this function:



Here is the caller graph for this function:



### 8.13.4 Variable Documentation

8.13.4.1 const char\* const devpath

Definition at line 8 of file devnode.h.

8.13.4.2 long gid

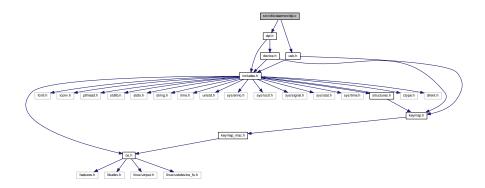
Definition at line 16 of file devnode.c.

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

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

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

Include dependency graph for dpi.c:



#### **Functions**

- void cmd dpi (usbdevice \*kb, usbmode \*mode, int dummy, const char \*stages, const char \*values)
- void cmd\_dpisel (usbdevice \*kb, usbmode \*mode, int dummy1, int dummy2, const char \*stage)
- void cmd\_lift (usbdevice \*kb, usbmode \*mode, int dummy1, int dummy2, const char \*height)
- void cmd\_snap (usbdevice \*kb, usbmode \*mode, int dummy1, int dummy2, const char \*enable)
- char \* printdpi (const dpiset \*dpi, const usbdevice \*kb)
- int updatedpi (usbdevice \*kb, int force)
- int savedpi (usbdevice \*kb, dpiset \*dpi, lighting \*light)
- int loaddpi (usbdevice \*kb, dpiset \*dpi, lighting \*light)

### 8.14.1 Function Documentation

8.14.1.1 void cmd\_dpi ( usbdevice \* kb, usbmode \* mode, int dummy, const char \* stages, const char \* values )

Definition at line 4 of file dpi.c.

References usbmode::dpi, DPI\_COUNT, dpiset::enabled, dpiset::x, and dpiset::y.

```
5
       int disable = 0;
       ushort x, y;
6
       // Try to scan X,Y values
if(sscanf(values, "%hu,%hu", &x, &y) != 2){
    // If that doesn't work, scan single number
    if(sscanf(values, "%hu", &x) == 1)
8
              y = x;

else if(!strncmp(values, "off", 3))

// If the right side says "off", disable the level(s)

disable = 1;
12
13
14
15
              else
16
                   // Otherwise, quit
17
18
19
         if((x == 0 || y == 0) && !disable)
20
               return:
         // Scan the left side for stage numbers (comma-separated)
21
         int left = strlen(stages);
22
         int position = 0, field = 0;
         char stagename[3];
         while (position < left && sscanf(stages + position, "%2[^,]%n", stagename, &field) == 1) {
25
26
              uchar stagenum;
              if(sscanf(stagename, "%hhu", &stagenum) && stagenum < DPI_COUNT){</pre>
27
28
                    // Set DPI for this stage
29
                    if (disable) {
30
                        mode->dpi.enabled &= ~(1 << stagenum);</pre>
                        mode->dpi.x[stagenum] = 0;
mode->dpi.y[stagenum] = 0;
31
32
33
                    } else {
34
                        mode->dpi.enabled |= 1 << stagenum;</pre>
                        mode->dpi.x[stagenum] = x;
```

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

Definition at line 44 of file dpi.c.

References dpiset::current, usbmode::dpi, and DPI\_COUNT.

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

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

Definition at line 53 of file dpi.c.

References usbmode::dpi, dpiset::lift, LIFT\_MAX, and LIFT\_MIN.

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

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

Definition at line 62 of file dpi.c.

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

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

Definition at line 152 of file dpi.c.

References lighting::b, ckb\_err, dpiset::current, DPI\_COUNT, dpiset::enabled, lighting::g, LED\_MOUSE, dpiset::lift, LIFT\_MAX, LIFT\_MIN, MSG\_SIZE, N\_MOUSE\_ZONES, lighting::r, dpiset::snap, usbrecv, dpiset::x, and dpiset::y.

Referenced by cmd\_hwload\_mouse().

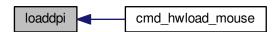
```
157
               { 0x0e, 0x13, 0x03, 1, },
158
               { 0x0e, 0x13, 0x04, 1, }
159
160
          uchar in_pkt[4][MSG_SIZE];
          for(int i = 0; i < 4; i++) {
    if(!usbrecv(kb, data_pkt[i], in_pkt[i]))</pre>
161
162
163
                    return -2;
164
               if (memcmp(in_pkt[i], data_pkt[i], 4)){
165
                    ckb_err("Bad input header\n");
166
                    return -3;
               }
167
168
          // Copy data from device
169
170
          dpi->enabled = in_pkt[0][4];
171
          dpi->enabled &= (1 << DPI_COUNT) - 1;
          dpi->current = in_pkt[1][4];
172
          if (dpi->current >= DPI_COUNT)
    dpi->current = 0;
173
174
175
          dpi->lift = in_pkt[2][4];
         if (dpi->lift < LIFT_MIN || dpi->lift > LIFT_MAX)
    dpi->lift = LIFT_MIN;
176
177
178
          dpi->snap = !!in_pkt[3][4];
179
          // Get X/Y DPIs
for(int i = 0; i < DPI_COUNT; i++) {</pre>
180
181
              uchar data_pkt[MSG_SIZE] = { 0x0e, 0x13, 0xd0, 1 };
182
183
               uchar in_pkt[MSG_SIZE];
               data_pkt[2] |= i;
184
185
               if(!usbrecv(kb, data_pkt, in_pkt))
186
                    return -2;
187
               if (memcmp(in_pkt, data_pkt, 4)) {
188
                    ckb_err("Bad input header\n");
189
190
               // Copy to profile

dpi->x[i] = *(ushort*)(in_pkt + 5);

dpi->y[i] = *(ushort*)(in_pkt + 7);

light->r[LED_MOUSE + N_MOUSE_ZONES + i] = in_pkt[9];

light->g[LED_MOUSE + N_MOUSE_ZONES + i] = in_pkt[10];
191
192
193
194
195
196
               light->b[LED_MOUSE + N_MOUSE_ZONES + i] = in_pkt[11];
197
          // Finished. Set SW DPI light to the current hardware level light->r[LED_MOUSE + 2] = light->r[LED_MOUSE +
198
199
       N_MOUSE_ZONES + dpi->current];
          light->g[LED_MOUSE + 2] = light->g[LED_MOUSE +
200
       N_MOUSE_ZONES + dpi->current];
201
          light->b[LED_MOUSE + 2] = light->b[LED_MOUSE +
       N_MOUSE_ZONES + dpi->current];
202
          return 0:
203 }
```



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

Definition at line 69 of file dpi.c.

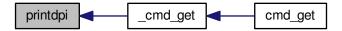
References \_readlines\_ctx::buffer, DPI\_COUNT, dpiset::enabled, dpiset::x, and dpiset::y.

Referenced by \_cmd\_get().

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

```
char* buffer = malloc(BUFFER_LEN);
       int length = 0;
for(int i = 0; i < DPI_COUNT; i++) {</pre>
74
            // Print the stage number
7.5
76
           int newlen = 0;
           snprintf(buffer + length, BUFFER_LEN - length, length == 0 ? "%d%n" : " %d%n", i, &newlen);
           length += newlen;
79
            // Print the DPI settings
80
            if(!(dpi->enabled & (1 << i)))</pre>
                snprintf(buffer + length, BUFFER_LEN - length, ":off%n", &newlen);
81
           else
82
               snprintf(buffer + length, BUFFER_LEN - length, ":%u,%u%n", dpi->x[i], dpi->
83
      y[i], &newlen);
84
            length += newlen;
85
86
       return buffer;
87 1
```

Here is the caller graph for this function:



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

Definition at line 124 of file dpi.c.

References lighting::b, dpiset::current, DPI\_COUNT, dpiset::enabled, lighting::g, LED\_MOUSE, dpiset::lift, MSG\_-SIZE, N\_MOUSE\_ZONES, lighting::r, dpiset::snap, usbsend, dpiset::x, and dpiset::y.

Referenced by cmd\_hwsave\_mouse().

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



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

Definition at line 89 of file dpi.c.

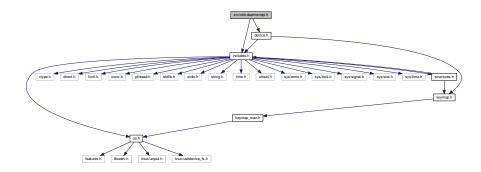
References usbdevice::active, dpiset::current, usbprofile::currentmode, usbmode::dpi, DPI\_COUNT, dpiset::enabled, dpiset::forceupdate, usbprofile::lastdpi, dpiset::lift, MSG\_SIZE, usbdevice::profile, dpiset::snap, usbsend, dpiset::x, and dpiset::y.

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

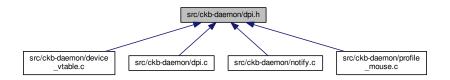
# 8.15 src/ckb-daemon/dpi.h File Reference

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

Include dependency graph for dpi.h:



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



## **Functions**

- int updatedpi (usbdevice \*kb, int force)
- int savedpi (usbdevice \*kb, dpiset \*dpi, lighting \*light)
- int loaddpi (usbdevice \*kb, dpiset \*dpi, lighting \*light)
- char \* printdpi (const dpiset \*dpi, const usbdevice \*kb)
- void cmd\_dpi (usbdevice \*kb, usbmode \*mode, int dummy, const char \*stages, const char \*values)
- void cmd dpisel (usbdevice \*kb, usbmode \*mode, int dummy1, int dummy2, const char \*stage)
- void cmd\_lift (usbdevice \*kb, usbmode \*mode, int dummy1, int dummy2, const char \*height)
- void cmd\_snap (usbdevice \*kb, usbmode \*mode, int dummy1, int dummy2, const char \*enable)

#### 8.15.1 Function Documentation

8.15.1.1 void cmd\_dpi ( usbdevice \* kb, usbmode \* mode, int dummy, const char \* stages, const char \* values )

Definition at line 4 of file dpi.c.

References usbmode::dpi, DPI\_COUNT, dpiset::enabled, dpiset::x, and dpiset::y.

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

```
18
      if((x == 0 || y == 0) && !disable)
20
          return;
      // Scan the left side for stage numbers (comma-separated)
2.1
2.2
      int left = strlen(stages);
      int position = 0, field = 0;
23
      char stagename[3];
25
      26
          if(sscanf(stagename, "%hhu", &stagenum) && stagenum < DPI_COUNT){</pre>
2.7
28
              // Set DPI for this stage
29
              if(disable){
                  mode->dpi.enabled &= ~(1 << stagenum);</pre>
30
31
                  mode->dpi.x[stagenum] = 0;
32
                  mode->dpi.y[stagenum] = 0;
33
                  mode->dpi.enabled |= 1 << stagenum;</pre>
34
                 mode->dpi.x[stagenum] = x;
mode->dpi.y[stagenum] = y;
35
36
38
39
          if(stages[position += field] == ',')
40
             position++;
41
```

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

Definition at line 44 of file dpi.c.

References dpiset::current, usbmode::dpi, and DPI\_COUNT.

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

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

Definition at line 53 of file dpi.c.

References usbmode::dpi, dpiset::lift, LIFT MAX, and LIFT MIN.

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

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

Definition at line 62 of file dpi.c.

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

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

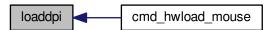
Definition at line 152 of file dpi.c.

References lighting::b, ckb\_err, dpiset::current, DPI\_COUNT, dpiset::enabled, lighting::g, LED\_MOUSE, dpiset::lift, LIFT\_MAX, LIFT\_MIN, MSG\_SIZE, N\_MOUSE\_ZONES, lighting::r, dpiset::snap, usbrecv, dpiset::x, and dpiset::y.

Referenced by cmd\_hwload\_mouse().

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

Here is the caller graph for this function:



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

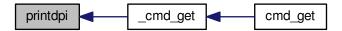
Definition at line 69 of file dpi.c.

References \_readlines\_ctx::buffer, DPI\_COUNT, dpiset::enabled, dpiset::x, and dpiset::y.

Referenced by \_cmd\_get().

```
69
70
       // Print all DPI settings
71
       const int BUFFER_LEN = 100;
       char* buffer = malloc(BUFFER_LEN);
       int length = 0;
73
74
       for(int i = 0; i < DPI_COUNT; i++) {</pre>
           // Print the stage number
int newlen = 0;
7.5
76
           snprintf(buffer + length, BUFFER_LEN - length, length == 0 ? "%d%n" : " %d%n", i, &newlen);
78
           length += newlen;
79
           // Print the DPI settings
80
           if(!(dpi->enabled & (1 << i)))</pre>
                snprintf(buffer + length, BUFFER_LEN - length, ":off%n", &newlen);
81
82
83
               snprintf(buffer + length, BUFFER_LEN - length, ":%u,%u%n", dpi->x[i], dpi->
      y[i], &newlen);
84
           length += newlen;
8.5
       return buffer;
86
87 }
```

Here is the caller graph for this function:



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

Definition at line 124 of file dpi.c.

References lighting::b, dpiset::current, DPI\_COUNT, dpiset::enabled, lighting::g, LED\_MOUSE, dpiset::lift, MSG\_-SIZE, N\_MOUSE\_ZONES, lighting::r, dpiset::snap, usbsend, dpiset::x, and dpiset::y.

Referenced by cmd hwsave mouse().

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

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

Here is the caller graph for this function:



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

Definition at line 89 of file dpi.c.

References usbdevice::active, dpiset::current, usbprofile::currentmode, usbmode::dpi, DPI\_COUNT, dpiset::enabled, dpiset::forceupdate, usbprofile::lastdpi, dpiset::lift, MSG\_SIZE, usbdevice::profile, dpiset::snap, usbsend, dpiset::x, and dpiset::y.

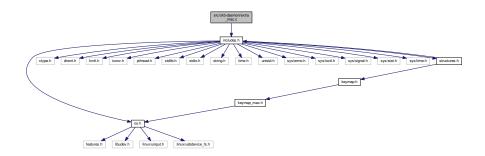
```
{
90
         if(!kb->active)
               return 0;
        dpiset* lastdpi = &kb->profile->lastdpi;
92
        dpiset* lastdpi = &kb->profile->currentmode->dpi;
// Don't do anything if the settings haven't changed
if(!force && !lastdpi->forceupdate && !newdpi->forceupdate
9.3
94
95
                  && !memcmp(lastdpi, newdpi, sizeof(dpiset)))
97
              return 0;
98
        lastdpi->forceupdate = newdpi->forceupdate = 0;
99
100
          // Send X/Y DPIs
          for(int i = 0; i < DPI_COUNT; i++) {</pre>
101
               uchar data_pkt[MSG_SIZE] = { 0x07, 0x13, 0xd0, 0 };
102
103
                data_pkt[2] |= i;
                *(ushort*)(data_pkt + 5) = newdpi->x[i];

*(ushort*)(data_pkt + 7) = newdpi->y[i];
104
105
                if(!usbsend(kb, data_pkt, 1))
106
                    return -1;
107
108
109
          // Send settings
110
          uchar data_pkt[4][MSG_SIZE] = {
111
               { 0x07, 0x13, 0x05, 0, newdpi->enabled }, 
{ 0x07, 0x13, 0x02, 0, newdpi->current }, 
{ 0x07, 0x13, 0x03, 0, newdpi->lift },
112
113
114
                { 0x07, 0x13, 0x04, 0, newdpi->snap, 0x05 }
116
117
          if(!usbsend(kb, data_pkt[0], 4))
118
          return -2;
// Finished
119
120
          memcpy(lastdpi, newdpi, sizeof(dpiset));
121
          return 0;
122 }
```

# 8.16 src/ckb-daemon/extra\_mac.c File Reference

#include "includes.h"

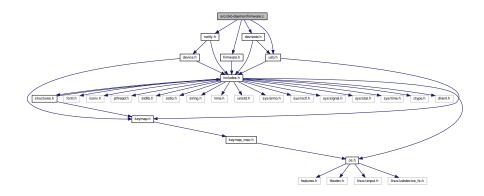
Include dependency graph for extra\_mac.c:



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

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

Include dependency graph for firmware.c:



# **Macros**

- #define FW\_OK 0
- #define FW\_NOFILE -1
- #define FW\_WRONGDEV -2
- #define FW\_USBFAIL -3
- #define FW\_MAXSIZE (255 \* 256)

# **Functions**

- int getfwversion (usbdevice \*kb)
- int fwupdate (usbdevice \*kb, const char \*path, int nnumber)
- int cmd\_fwupdate (usbdevice \*kb, usbmode \*dummy1, int nnumber, int dummy2, const char \*path)

#### 8.17.1 Macro Definition Documentation

```
8.17.1.1 #define FW_MAXSIZE (255 * 256)
```

Definition at line 51 of file firmware.c.

Referenced by fwupdate().

#### 8.17.1.2 #define FW\_NOFILE -1

Definition at line 7 of file firmware.c.

Referenced by cmd\_fwupdate(), and fwupdate().

#### 8.17.1.3 #define FW\_OK 0

Definition at line 6 of file firmware.c.

Referenced by cmd\_fwupdate(), and fwupdate().

#### 8.17.1.4 #define FW\_USBFAIL -3

Definition at line 9 of file firmware.c.

Referenced by cmd\_fwupdate(), and fwupdate().

#### 8.17.1.5 #define FW\_WRONGDEV -2

Definition at line 8 of file firmware.c.

Referenced by cmd\_fwupdate(), and fwupdate().

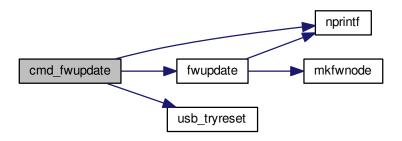
# 8.17.2 Function Documentation

8.17.2.1 int cmd\_fwupdate ( usbdevice \* kb, usbmode \* dummy1, int nnumber, int dummy2, const char \* path )

Definition at line 154 of file firmware.c.

References FEAT\_FWUPDATE, FW\_NOFILE, FW\_OK, FW\_USBFAIL, FW\_WRONGDEV, fwupdate(), HAS\_FEATURES, nprintf(), and usb\_tryreset().

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



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

Definition at line 55 of file firmware.c.

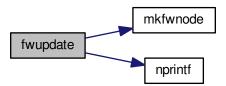
References ckb\_err, ckb\_info, FW\_MAXSIZE, FW\_NOFILE, FW\_OK, FW\_USBFAIL, FW\_WRONGDEV, usbdevice::fwversion, mkfwnode(), MSG\_SIZE, nprintf(), usbdevice::product, usbdevice::usbdelay, usbsend, and usbdevice::vendor.

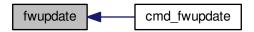
Referenced by cmd fwupdate().

```
{
56
        // Read the firmware from the given path
57
        char* fwdata = calloc(1, FW_MAXSIZE + 256);
        int fd = open(path, O_RDONLY);
58
        if (fd == -1) {
59
            ckb_err("Failed to open firmware file %s: %s\n", path, strerror(errno));
60
            return FW_NOFILE;
62
63
        ssize_t length = read(fd, fwdata, FW_MAXSIZE + 1);
       if(length <= 0x108 || length > FW_MAXSIZE) {
64
            ckb_err("Failed to read firmware file %s: %s\n", path, length <= 0 ? strerror(errno) : "
65
       Wrong size");
66
            close(fd);
            return FW_NOFILE;
68
        close(fd):
69
70
        short vendor, product, version;
72
        // Copy the vendor ID, product ID, and version from the firmware file
        memcpy(&vendor, fwdata + 0x102, 2);
        memcpy(&product, fwdata + 0x104, 2);
memcpy(&version, fwdata + 0x106, 2);
74
7.5
76
        // Check against the actual device
        if(vendor != kb->vendor || product != kb->product) {
77
78
            ckb_err("Firmware file %s doesn't match device (V: %04x P: %04x)\n", path, vendor, product);
79
             return FW_WRONGDEV;
80
        ckb_info("Loading firmware version %04x from %s\n", version, path);
nprintf(kb, nnumber, 0, "fwupdate %s 0/%d\n", path, (int)length);
// Force the device to 10ms delay (we need to deliver packets very slowly to make sure it doesn't get
81
82
83
        overwhelmed)
84
        kb->usbdelay = 10;
85
        // Send the firmware messages (256 bytes at a time)
86
        uchar data_pkt[7][MSG_SIZE] =
            { 0x07, 0x0c, 0xf0, 0x01, 0 },
{ 0x07, 0x0d, 0xf0, 0 },
87
88
             { 0x7f, 0x01, 0x3c, 0 },
             { 0x7f, 0x02, 0x3c, 0 },
90
             { 0x7f, 0x03, 0x3c, 0 },
92
             { 0x7f, 0x04, 0x3c, 0 },
9.3
             \{ 0x7f, 0x05, 0x10, 0 \}
94
       };
95
        int output = 0, last = 0;
        int index = 0;
```

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





#### 8.17.2.3 int getfwversion ( usbdevice \* kb )

Definition at line 11 of file firmware.c.

References ckb\_err, ckb\_warn, FEAT\_POLLRATE, usbdevice::features, usbdevice::fwversion, MSG\_SIZE, usbdevice::pollrate, usbdevice::product, usbrecv, and usbdevice::vendor.

Referenced by \_start\_dev().

```
11
       // Ask board for firmware info
12
       uchar data_pkt[MSG_SIZE] = { 0x0e, 0x01, 0 };
13
       uchar in_pkt[MSG_SIZE];
14
15
       if(!usbrecv(kb, data_pkt, in_pkt))
       return -1;
if(in_pkt[0] != 0x0e || in_pkt[1] != 0x01){
    ckb_err("Bad input header\n");
16
17
18
19
           return -1;
20
       short vendor, product, version, bootloader;
       // Copy the vendor ID, product ID, version, and poll rate from the firmware data
23
       memcpy(&version, in_pkt + 8, 2);
       memcpy(&bootloader, in_pkt + 10, 2);
24
       memcpy(&vendor, in_pkt + 12, 2);
memcpy(&product, in_pkt + 14, 2);
25
26
       char poll = in_pkt[16];
       if (pol1 <= 0) {
    pol1 = -1;
28
29
           kb->features &= ~FEAT_POLLRATE;
30
31
       // Print a warning if the message didn't match the expected data
32
       if (vendor != kb->vendor)
33
34
           ckb_warn("Got vendor ID 04x (expected 04x)\n", vendor, kb->
      vendor);
      35
36
      product);
37
       // Set firmware version and poll rate
38
       if(version == 0 || bootloader == 0){
39
            // Needs firmware update
40
            kb \rightarrow fwversion = 0;
           kb->pollrate = -1;
41
       } else {
42
43
           if(version != kb->fwversion && kb->fwversion != 0)
               ckb_warn("Got firmware version %04x (expected %04x)\n", version, kb->
44
      fwversion);
           kb->fwversion = version;
kb->pollrate = poll;
45
46
47
48
       return 0;
49 }
```

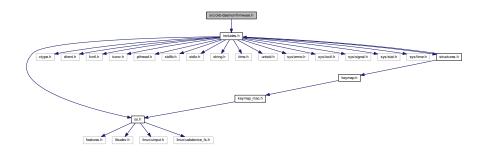
Here is the caller graph for this function:



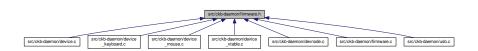
# 8.18 src/ckb-daemon/firmware.h File Reference

#include "includes.h"

Include dependency graph for firmware.h:



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



## **Functions**

- int getfwversion (usbdevice \*kb)
- int cmd\_fwupdate (usbdevice \*kb, usbmode \*dummy1, int nnumber, int dummy2, const char \*path)

#### 8.18.1 Function Documentation

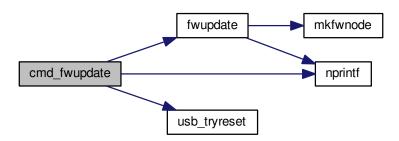
8.18.1.1 int cmd\_fwupdate ( usbdevice \* kb, usbmode \* dummy1, int nnumber, int dummy2, const char \* path )

Definition at line 154 of file firmware.c.

References FEAT\_FWUPDATE, FW\_NOFILE, FW\_OK, FW\_USBFAIL, FW\_WRONGDEV, fwupdate(), HAS\_FEATURES, nprintf(), and usb\_tryreset().

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

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



# 8.18.1.2 int getfwversion ( usbdevice \* kb )

Definition at line 11 of file firmware.c.

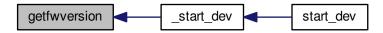
References ckb\_err, ckb\_warn, FEAT\_POLLRATE, usbdevice::features, usbdevice::fwversion, MSG\_SIZE, usbdevice::pollrate, usbdevice::product, usbrecv, and usbdevice::vendor.

Referenced by \_start\_dev().

```
11
        // Ask board for firmware info
12
13
        uchar data_pkt[MSG_SIZE] = { 0x0e, 0x01, 0 };
14
        uchar in_pkt[MSG_SIZE];
1.5
        if(!usbrecv(kb, data_pkt, in_pkt))
        return -1;
if(in_pkt[0] != 0x0e || in_pkt[1] != 0x01){
16
17
            ckb_err("Bad input header\n");
18
19
             return -1;
20
21
        short vendor, product, version, bootloader;
        // Copy the vendor ID, product ID, version, and poll rate from the firmware data
memopy(&version, in_pkt + 8, 2);
memopy(&bootloader, in_pkt + 10, 2);
22
23
25
        memcpy(&vendor, in_pkt + 12, 2);
        memcpy(&product, in_pkt + 14, 2);
27
        char poll = in_pkt[16];
        if (poll <= 0) {
   poll = -1;
2.8
29
30
             kb->features &= ~FEAT_POLLRATE;
        }
```

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

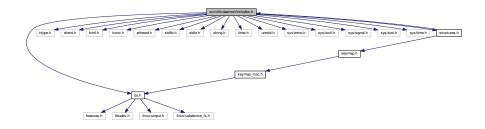
Here is the caller graph for this function:



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

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

Include dependency graph for includes.h:



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



#### **Macros**

- #define INDEX OF(entry, array) (int)(entry array)
- #define ckb\_s\_out stdout
- #define ckb s err stdout
- #define \_\_FILE\_NOPATH\_\_ (strrchr(\_\_FILE\_\_, '/') ? strrchr(\_\_FILE\_\_, '/') + 1 : \_\_FILE\_\_)
- #define ckb fatal nofile(fmt, args...) fprintf(ckb s err, "[F] " fmt, ## args)
- #define ckb\_fatal\_fn(fmt, file, line, args...) fprintf(ckb\_s\_err, "[F] %s (via %s:%d): " fmt, \_\_func\_\_, file, line, ## args)
- #define ckb\_fatal(fmt, args...) fprintf(ckb\_s\_err, "[F] %s (%s:%d): " fmt, \_\_func\_\_, \_\_FILE\_NOPATH\_\_, \_\_L-INE\_\_, ## 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\_\_, \_\_LI-NE\_\_, ## args)
- #define ckb warn nofile(fmt, args...) fprintf(ckb s out, "[W] " fmt, ## args)
- #define ckb\_warn\_fn(fmt, file, line, args...) fprintf(ckb\_s\_out, "[W] %s (via %s:%d): " fmt, \_\_func\_\_, file, line, ## args)
- #define ckb\_warn(fmt, args...) fprintf(ckb\_s\_out, "[W] %s (%s:%d): " fmt, \_\_func\_\_, \_\_FILE\_NOPATH\_\_,
   \_\_LINE\_\_, ## args)
- #define ckb\_info\_nofile(fmt, args...) fprintf(ckb\_s\_out, "[I] " fmt, ## args)
- #define ckb\_info\_fn(fmt, file, line, args...) fprintf(ckb\_s\_out, "[I] " fmt, ## args)
- #define ckb\_info(fmt, args...) fprintf(ckb\_s\_out, "[I] " fmt, ## args)
- #define timespec\_gt(left, right) ((left).tv\_sec > (right).tv\_sec || ((left).tv\_sec == (right).tv\_sec && (left).tv\_nsec > (right).tv\_nsec))
- #define timespec\_eq(left, right) ((left).tv\_sec == (right).tv\_sec && (left).tv\_nsec == (right).tv\_nsec)
- #define timespec\_ge(left, right) ((left).tv\_sec > (right).tv\_sec || ((left).tv\_sec == (right).tv\_sec && (left).tv\_nsec >= (right).tv\_nsec))
- #define timespec lt(left, right) (!timespec ge(left, right))
- #define timespec\_le(left, right) (!timespec\_gt(left, right))

# **Typedefs**

- · typedef unsigned char uchar
- · typedef unsigned short ushort

# **Functions**

void timespec\_add (struct timespec \*timespec, long nanoseconds)

#### 8.19.1 Macro Definition Documentation

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

Definition at line 40 of file includes.h.

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

Definition at line 49 of file includes.h.

Referenced by \_mkdevpath(), fwupdate(), getfwversion(), loaddpi(), loadrgb\_kb(), loadrgb\_mouse(), os\_sendindicators(), os setupusb(), restart(), setupusb(), uinputopen(), usb tryreset(), and usbadd().

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

Definition at line 48 of file includes.h.

Referenced by \_nk95cmd(), \_usbrecv(), os\_usbrecv(), and os\_usbsend().

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

Definition at line 47 of file includes.h.

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

Definition at line 46 of file includes.h.

Referenced by usbmain().

8.19.1.6 #define ckb\_fatal\_fn( fmt, file, line, args... ) fprintf(ckb\_s\_err, "[F] %s (via %s:%d): " fmt, \_\_func\_\_, file, line, ## args)

Definition at line 45 of file includes.h.

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

Definition at line 44 of file includes.h.

Referenced by main().

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

Definition at line 55 of file includes.h.

Referenced by \_setupusb(), \_start\_dev(), closeusb(), cmd\_restart(), fwupdate(), main(), os\_inputmain(), os\_setupusb(), quitWithLock(), rmdevpath(), and usb\_tryreset().

8.19.1.9 #define ckb\_info\_fn( fmt, file, line, args... ) fprintf(ckb\_s\_out, "[I] " fmt, ## args)

Definition at line 54 of file includes.h.

8.19.1.10 #define ckb\_info\_nofile( fmt, args... ) fprintf(ckb\_s\_out, "[I] " fmt, ## args)

Definition at line 53 of file includes.h.

Referenced by main().

8.19.1.11 #define ckb\_s\_err stdout

Definition at line 36 of file includes.h.

8.19.1.12 #define ckb\_s\_out stdout

Definition at line 35 of file includes.h.

8.19.1.13 #define ckb\_warn( fmt, args... ) fprintf(ckb\_s\_out, "[W] %s (%s:%d): " fmt, \_\_func\_\_, \_\_FILE\_NOPATH\_\_, \_\_LINE\_\_, ## args)

Definition at line 52 of file includes.h.

Referenced by \_mkdevpath(), \_mknotifynode(), \_start\_dev(), \_updateconnected(), getfwversion(), hid\_kb\_translate(), isync(), mkfwnode(), os\_inputclose(), os\_keypress(), os\_mousemove(), os\_setupusb(), readlines(), rmdevpath(), uinputopen(), and usbmain().

8.19.1.14 #define ckb\_warn\_fn( fmt, file, line, args... ) fprintf(ckb\_s\_out, "[W] %s (via %s:%d): " fmt, \_\_func\_\_, file, line, ## args)

Definition at line 51 of file includes.h.

Referenced by os\_usbrecv(), and os\_usbsend().

8.19.1.15 #define ckb\_warn\_nofile( fmt, args... ) fprintf(ckb\_s\_out, "[W] " fmt, ## args)

Definition at line 50 of file includes.h.

Referenced by main().

8.19.1.16 #define INDEX\_OF( entry, array ) (int)(entry - array)

Definition at line 27 of file includes.h.

Referenced by \_mkdevpath(), \_mknotifynode(), \_rmnotifynode(), \_setupusb(), closeusb(), mkfwnode(), nprintf(), os\_closeusb(), os\_inputmain(), os\_inputopen(), os\_setupusb(), readcmd(), and rmdevpath().

8.19.1.17 #define timespec\_eq( left, right ) ((left).tv\_sec == (right).tv\_sec && (left).tv\_nsec == (right).tv\_nsec)

Definition at line 60 of file includes.h.

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

Definition at line 61 of file includes.h.

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

Definition at line 59 of file includes.h.

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

Definition at line 63 of file includes.h.

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

Definition at line 62 of file includes.h.

# 8.19.2 Typedef Documentation

8.19.2.1 typedef unsigned char uchar

Definition at line 24 of file includes.h.

#### 8.19.2.2 typedef unsigned short ushort

Definition at line 25 of file includes.h.

# 8.19.3 Function Documentation

8.19.3.1 void timespec\_add ( struct timespec \* timespec, long nanoseconds )

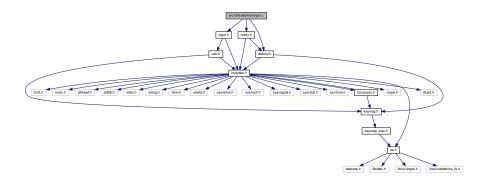
Definition at line 19 of file main.c.

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

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

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

Include dependency graph for input.c:



# **Macros**

#define IS\_WHEEL(scan, kb) (((scan) == KEY\_VOLUMEUP || (scan) == KEY\_VOLUMEDOWN || (scan) == BTN\_WHEELUP || (scan) == BTN\_WHEELDOWN) && !!S\_K65(kb))

#### **Functions**

- int macromask (const uchar \*key1, const uchar \*key2)
- static void inputupdate keys (usbdevice \*kb)
- void inputupdate (usbdevice \*kb)
- void updateindicators\_kb (usbdevice \*kb, int force)
- void initbind (binding \*bind)
- void freebind (binding \*bind)
- void cmd\_bind (usbdevice \*kb, usbmode \*mode, int dummy, int keyindex, const char \*to)
- void cmd\_unbind (usbdevice \*kb, usbmode \*mode, int dummy, int keyindex, const char \*to)
- void cmd rebind (usbdevice \*kb, usbmode \*mode, int dummy, int keyindex, const char \*to)
- static void \_cmd\_macro (usbmode \*mode, const char \*keys, const char \*assignment)
- void cmd\_macro (usbdevice \*kb, usbmode \*mode, const int notifynumber, const char \*keys, const char \*assignment)

#### 8.20.1 Macro Definition Documentation

8.20.1.1 #define IS\_WHEEL( scan, kb ) (((scan) == KEY\_VOLUMEUP || (scan) == KEY\_VOLUMEDOWN || (scan) == BTN WHEELUP || (scan) == BTN WHEELDOWN) && !IS K65(kb))

Referenced by inputupdate keys().

#### 8.20.2 Function Documentation

**8.20.2.1** static void \_cmd\_macro ( usbmode \* mode, const char \* keys, const char \* assignment ) [static]

Definition at line 226 of file input.c.

References keymacro::actioncount, keymacro::actions, usbmode::bind, keymacro::combo, macroaction::down, keymap, MACRO\_MAX, binding::macrocap, binding::macrocount, binding::macros, N\_KEYBYTES\_INPUT, N\_KE-YS\_INPUT, macroaction::scan, key::scan, and SET\_KEYBIT.

Referenced by cmd\_macro().

```
226
                                                                                             {
227
         binding* bind = &mode->bind;
228
         if(!keys && !assignment){
             // Null strings = "macro clear" -> erase the whole thing
for(int i = 0; i < bind->macrocount; i++)
229
230
                  free(bind->macros[i].actions);
231
232
             bind->macrocount = 0;
233
234
235
         if (bind->macrocount >= MACRO MAX)
236
             return;
         // Create a key macro
237
238
         keymacro macro;
239
         memset(&macro, 0, sizeof(macro));
240
         // Scan the left side for key names, separated by +
         int empty = 1;
int left = strlen(keys), right = strlen(assignment);
int position = 0, field = 0;
241
242
243
         char keyname[12];
244
245
         while (position < left && sscanf(keys + position, "%10[^+] %n", keyname, &field) == 1) {</pre>
246
             int keycode;
247
             if((sscanf(keyname, "#%d", &keycode) && keycode >= 0 && keycode <
      N_KEYS_INPUT)
                        || (sscanf(keyname, "#x%x", &keycode) && keycode >= 0 && keycode <
248
      N_KEYS_INPUT)){
249
                 // Set a key numerically
250
                  SET_KEYBIT(macro.combo, keycode);
251
                  empty = 0;
             2.52
253
254
255
                      if(keymap[i].name && !strcmp(keyname, keymap[i].name)){
256
                           macro.combo[i / 8] |= 1 << (i % 8);
                           empty = 0;
2.57
258
                           break;
259
                      }
260
                 }
261
262
             if (keys[position += field] == '+')
263
                 position++;
264
         if (empty)
2.65
266
             return;
         // Count the number of actions (comma separated)
267
         int count = 1;
268
269
         for(const char* c = assignment; *c != 0; c++){
270
             if(*c == ',')
271
                 count++;
272
273
         // Allocate a buffer for them
274
         macro.actions = calloc(count, sizeof(macroaction));
275
         macro.actioncount = 0;
276
         // Scan the actions
277
         position = 0;
278
         field = 0:
         while(position < right && sscanf(assignment + position, "%11[^,]%n", keyname, &field) == 1) {
   if(!strcmp(keyname, "clear"))</pre>
279
280
281
                  break;
282
             int down = (\text{keyname}[0] == '+');
             if(down || keyname[0] == '-'){
283
284
                  int keycode;
                  if((sscanf(keyname + 1, "#%d", &keycode) && keycode >= 0 && keycode < N_KEYS_INPUT)

|| (sscanf(keyname + 1, "#x%x", &keycode) && keycode >= 0 && keycode <
285
286
       N KEYS INPUT)){
287
                      // Set a key numerically
288
                      macro.actions[macro.actioncount].scan =
       keymap[keycode].scan;
289
                      macro.actions[macro.actioncount].down = down;
290
                      macro.actioncount++;
291
                  } else {
                      // Find this key in the keymap
for(unsigned i = 0; i < N_KEYS_INPUT; i++){</pre>
292
293
                           if(keymap[i].name && !strcmp(keyname + 1, keymap[i].name)){
294
295
                               macro.actions[macro.actioncount].scan =
       keymap[i].scan;
296
                               macro.actions[macro.actioncount].down = down;
297
                               macro.actioncount++;
298
                               break;
299
                           }
300
                      }
301
302
303
             if(assignment[position += field] == ',')
304
                  position++;
305
         }
306
307
         // See if there's already a macro with this trigger
```

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



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

Definition at line 188 of file input.c.

References binding::base, usbmode::bind, imutex, keymap, N\_KEYS\_INPUT, and key::scan.

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

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

Definition at line 332 of file input.c.

References \_cmd\_macro(), and imutex.

Here is the call graph for this function:



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

Definition at line 218 of file input.c.

References binding::base, usbmode::bind, imutex, keymap, N\_KEYS\_INPUT, and key::scan.

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

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

Definition at line 210 of file input.c.

References binding::base, usbmode::bind, imutex, KEY UNBOUND, and N KEYS INPUT.

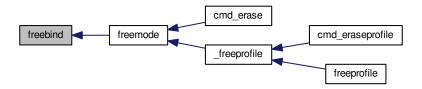
```
210
211    if(keyindex >= N_KEYS_INPUT)
212        return;
213    pthread_mutex_lock(imutex(kb));
214    mode->bind.base[keyindex] = KEY_UNBOUND;
215    pthread_mutex_unlock(imutex(kb));
216 }
```

8.20.2.6 void freebind (binding \* bind)

Definition at line 181 of file input.c.

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

Referenced by freemode().



# 8.20.2.7 void initbind ( binding \* bind )

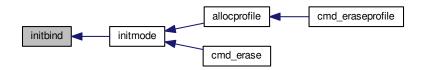
Definition at line 173 of file input.c.

References binding::base, keymap, binding::macrocap, binding::macrocount, binding::macros, N\_KEYS\_INPUT, and key::scan.

Referenced by initmode().

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

Here is the caller graph for this function:



# 8.20.2.8 void inputupdate ( usbdevice \*kb )

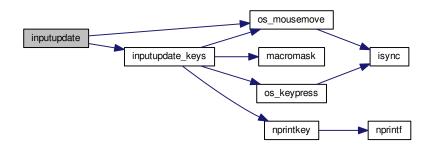
Definition at line 122 of file input.c.

References usbdevice::input, inputupdate\_keys(), os\_mousemove(), usbdevice::profile, usbinput::rel\_x, usbinput::rel\_y, usbdevice::uinput\_kb, and usbdevice::uinput\_mouse.

Referenced by os\_inputmain(), setactive\_kb(), and setactive\_mouse().

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

Here is the call graph for this function:



Here is the caller graph for this function:



**8.20.2.9** static void inputupdate\_keys ( usbdevice \* kb ) [static]

Definition at line 15 of file input.c.

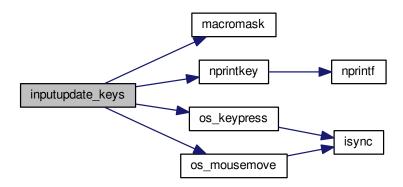
References keymacro::actioncount, keymacro::actions, usbdevice::active, binding::base, usbmode::bind, keymacro::combo, usbprofile::currentmode, usbdevice::delay, macroaction::down, usbdevice::input, IS\_MOD, IS\_WHEEL, keymap, usbinput::keys, binding::macrocount, macromask(), binding::macros, N\_KEYBYTES\_INPUT, N\_KEY-S\_INPUT, usbmode::notify, nprintkey(), os\_keypress(), os\_mousemove(), OUTFIFO\_MAX, usbinput::prevkeys, usbdevice::profile, macroaction::rel\_x, macroaction::rel\_y, macroaction::scan, key::scan, SCAN\_SILENT, and keymacro::triggered.

Referenced by inputupdate().

```
15
       usbmode* mode = kb->profile->currentmode;
16
       binding* bind = &mode->bind;
17
       usbinput * input = &kb->input;
18
       // Don't do anything if the state hasn't changed
       if(!memcmp(input->prevkeys, input->keys, N_KEYBYTES_INPUT))
21
            return;
       \ensuremath{//} Look for macros matching the current state
2.2
23
       int macrotrigger = 0;
24
       if (kb->active) {
            for(int i = 0; i < bind->macrocount; i++) {
```

```
26
                 keymacro* macro = &bind->macros[i];
                 if (macromask(input->keys, macro->combo)) {
28
                      if (!macro->triggered) {
2.9
                           macrotrigger = 1;
                           macro->triggered = 1;
30
                           // Send events for each keypress in the macro
31
                           for(int a = 0; a < macro->actioncount; a++) {
32
                               macroaction* action = macro->actions + a;
33
                                if(action->rel_x != 0 || action->rel_y != 0)
34
35
                                    os_mousemove(kb, action->rel_x, action->
       rel_y);
36
                               else {
                                    os_keypress(kb, action->scan, action->
37
       down);
38
                                    if (kb->delay) {
                                         if (a > 200) usleep (100);
else if (a > 20) usleep(30);
39
40
41
42
44
45
                 } else {
46
                      macro->triggered = 0;
47
48
            }
49
        // Make a list of keycodes to send. Rearrange them so that modifier keydowns always come first
50
51
        // and modifier keyups always come last. This ensures that shortcut keys will register properly
52
        // even if both keydown events happen at once.
53
        // N_KEYS + 4 is used because the volume wheel generates keydowns and keyups at the same time
        // (it's currently impossible to press all four at once, but safety first) int events[N_KEYS_INPUT + 4];
54
55
        int modcount = 0, keycount = 0, rmodcount = 0;
57
        for(int byte = 0; byte < N_KEYBYTES_INPUT; byte++){</pre>
             char oldb = input->prevkeys[byte], newb = input->keys[byte];
58
             if(oldb == newb)
59
                 continue;
60
             for(int bit = 0; bit < 8; bit++){</pre>
61
                 int keyindex = byte * 8 + bit;
                 if (keyindex >= N_KEYS_INPUT)
63
64
                      break;
                 const key* map = keymap + keyindex;
int scancode = (kb->active) ? bind->base[keyindex] : map->
6.5
66
       scan;
67
                 char mask = 1 << bit;</pre>
68
                 char old = oldb & mask, new = newb & mask;
69
                 // If the key state changed, send it to the input device
70
                 if(old != new){
                      // Don't echo a key press if a macro was triggered or if there's no scancode associated if(!macrotrigger && !(scancode & SCAN_SILENT)) {
71
72
73
                           if (IS_MOD (scancode)) {
74
                                if (new) {
7.5
                                    // Modifier down: Add to the end of modifier keys
                                    for(int i = keycount + rmodcount; i > 0; i--)
    events[modcount + i] = events[modcount + i - 1];
76
77
                                    // Add 1 to the scancode because A is zero on OSX
78
                                    // Positive code = keydown, negative code = keyup
                                    events[modcount++] = scancode + 1;
80
                               81
82
                                    events[modcount + keycount + rmodcount++] = -(scancode + 1);
8.3
84
                               }
85
                           } else {
                               // Regular keypress: add to the end of regular keys
87
                                for(int i = rmodcount; i > 0; i--)
                               events[modcount + keycount + i] = events[modcount + keycount + i - 1];
events[modcount + keycount++] = new ? (scancode + 1) : -(scancode + 1);
88
89
                                // The volume wheel and the mouse wheel don't generate keyups, so create them
90
        automatically
91
   #define IS_WHEEL(scan, kb)
                                    (((scan) == KEY_VOLUMEUP || (scan) == KEY_VOLUMEDOWN || (scan) == BTN_WHEELUP
        || (scan) == BTN_WHEELDOWN) && !IS_K65(kb))
92
                                if(new && IS_WHEEL(map->scan, kb)){
                                    for(int i = rmodcount; i > 0; i--)
    events[modcount + keycount + i] = events[modcount + keycount + i - 1];
events[modcount + keycount++] = -(scancode + 1);
93
94
95
                                    input->keys[byte] &= ~mask;
96
97
98
                           }
99
                       // Print notifications if desired
100
101
                       if(kb->active){
102
                            for(int notify = 0; notify < OUTFIFO_MAX; notify++){</pre>
                                 if (mode->notify[notify][byte] & mask) {
103
104
                                     nprintkey(kb, notify, keyindex, new);
                                     // Wheels doesn't generate keyups
if(new && IS_WHEEL(map->scan, kb))
    nprintkey(kb, notify, keyindex, 0);
105
106
107
```

Here is the call graph for this function:



Here is the caller graph for this function:



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

Definition at line 5 of file input.c.

References N\_KEYBYTES\_INPUT.

Referenced by inputupdate\_keys().



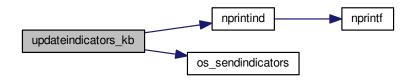
# 8.20.2.11 void updateindicators\_kb ( usbdevice \* kb, int force )

Definition at line 142 of file input.c.

References usbdevice::active, usbprofile::currentmode, DELAY\_SHORT, usbdevice::hw\_ileds, usbdevice::hw\_ileds\_old, I\_CAPS, I\_NUM, I\_SCROLL, usbdevice::ileds, usbmode::iontify, usbmode::ioff, usbmode::ion, nprintind(), os\_sendindicators(), OUTFIFO\_MAX, and usbdevice::profile.

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

Here is the call graph for this function:

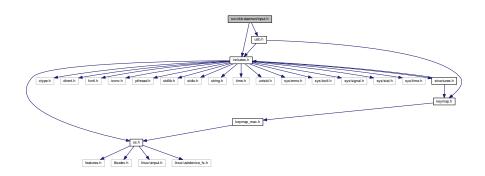


# 8.21 src/ckb-daemon/input.h File Reference

#include "includes.h"

#include "usb.h"

Include dependency graph for input.h:



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



#### **Macros**

#define IS\_MOD(s) ((s) == KEY\_CAPSLOCK || (s) == KEY\_NUMLOCK || (s) == KEY\_SCROLLLOCK || (s) == KEY\_LEFTSHIFT || (s) == KEY\_RIGHTSHIFT || (s) == KEY\_LEFTCTRL || (s) == KEY\_RIGHTCTRL || (s) == KEY\_LEFTMETA || (s) == KEY\_RIGHTMETA || (s) == KEY\_FN)

#### **Functions**

- int os\_inputopen (usbdevice \*kb)
- void os\_inputclose (usbdevice \*kb)
- void inputupdate (usbdevice \*kb)
- void updateindicators kb (usbdevice \*kb, int force)
- void initbind (binding \*bind)
- void freebind (binding \*bind)
- void cmd\_bind (usbdevice \*kb, usbmode \*mode, int dummy, int keyindex, const char \*to)
- void cmd\_unbind (usbdevice \*kb, usbmode \*mode, int dummy, int keyindex, const char \*ignored)
- void cmd\_rebind (usbdevice \*kb, usbmode \*mode, int dummy, int keyindex, const char \*ignored)
- void cmd\_macro (usbdevice \*kb, usbmode \*mode, const int notifynumber, const char \*keys, const char \*assignment)
- void os\_keypress (usbdevice \*kb, int scancode, int down)
- void os\_mousemove (usbdevice \*kb, int x, int y)
- int os setupindicators (usbdevice \*kb)

# 8.21.1 Macro Definition Documentation

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

Definition at line 34 of file input.h.

Referenced by inputupdate\_keys().

#### 8.21.2 Function Documentation

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

Definition at line 188 of file input.c.

References binding::base, usbmode::bind, imutex, keymap, N\_KEYS\_INPUT, and key::scan.

```
188
        if (keyindex >= N_KEYS_INPUT)
189
             return;
191
         // Find the key to bind to
        int tocode = 0;

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

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

Definition at line 332 of file input.c.

References \_cmd\_macro(), and imutex.

Here is the call graph for this function:



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

Definition at line 218 of file input.c.

References binding::base, usbmode::bind, imutex, keymap, N\_KEYS\_INPUT, and key::scan.

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

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

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

Definition at line 210 of file input.c.

References binding::base, usbmode::bind, imutex, KEY UNBOUND, and N KEYS INPUT.

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

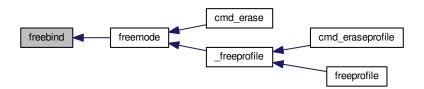
#### 8.21.2.5 void freebind (binding \* bind)

Definition at line 181 of file input.c.

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

Referenced by freemode().

Here is the caller graph for this function:



# 8.21.2.6 void initbind (binding \* bind)

Definition at line 173 of file input.c.

References binding::base, keymap, binding::macrocap, binding::macrocount, binding::macros, N\_KEYS\_INPUT, and key::scan.

Referenced by initmode().



# 8.21.2.7 void inputupdate ( usbdevice \* kb )

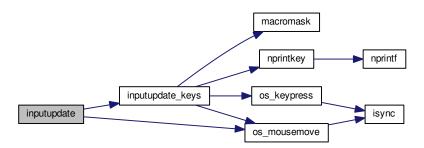
Definition at line 122 of file input.c.

References usbdevice::input, inputupdate\_keys(), os\_mousemove(), usbdevice::profile, usbinput::rel\_x, usbinput::rel\_y, usbdevice::uinput\_kb, and usbdevice::uinput\_mouse.

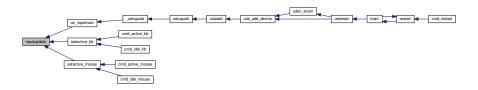
Referenced by os\_inputmain(), setactive\_kb(), and setactive\_mouse().

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

Here is the call graph for this function:



Here is the caller graph for this function:



#### 8.21.2.8 void os\_inputclose ( usbdevice \* kb )

Definition at line 70 of file input\_linux.c.

References ckb\_warn, usbdevice::uinput\_kb, and usbdevice::uinput\_mouse.

Referenced by closeusb().

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

Here is the caller graph for this function:



# 8.21.2.9 int os\_inputopen ( usbdevice \* kb )

Definition at line 49 of file input linux.c.

References usbdevice::fwversion, INDEX\_OF, keyboard, usbdevice::name, usbdevice::product, usbdevice::uinput\_kb, usbdevice::uinput\_mouse, uinputopen(), and usbdevice::vendor.

Referenced by \_setupusb().

```
49
        // Create the new input device
51
        int index = INDEX_OF(kb, keyboard);
52
        struct uinput_user_dev indev;
        memset(&indev, 0, sizeof(indev));
snprintf(indev.name, UINPUT_MAX_NAME_SIZE, "ckb%d: %s", index, kb->name);
indev.id.bustype = BUS_USB;
5.3
54
55
        indev.id.vendor = kb->vendor;
57
        indev.id.product = kb->product;
58
        indev.id.version = kb->fwversion;
        // Open keyboard
int fd = uinputopen(&indev, 0);
59
60
        kb->uinput_kb = fd;
61
        if(fd <= 0)
63
             return 0;
        // Open mouse
        fd = uinputopen(&indev, 1);
kb->uinput_mouse = fd;
65
66
        return fd <= 0;
67
68 }
```



Here is the caller graph for this function:



8.21.2.10 void os\_keypress ( usbdevice \* kb, int scancode, int down )

Definition at line 112 of file input linux.c.

References BTN\_WHEELDOWN, BTN\_WHEELUP, ckb\_warn, isync(), SCAN\_MOUSE, usbdevice::uinput\_kb, and usbdevice::uinput\_mouse.

Referenced by inputupdate\_keys().

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

Here is the call graph for this function:



Here is the caller graph for this function:



# 8.21.2.11 void os\_mousemove ( usbdevice \*kb, int x, int y )

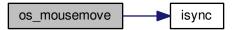
Definition at line 137 of file input\_linux.c.

References ckb\_warn, isync(), and usbdevice::uinput\_mouse.

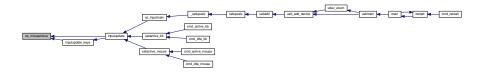
Referenced by inputupdate(), and inputupdate\_keys().

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

Here is the call graph for this function:



Here is the caller graph for this function:



# 8.21.2.12 int os\_setupindicators ( usbdevice \* kb )

Definition at line 183 of file input\_linux.c.

References \_ledthread(), usbdevice::hw\_ileds, usbdevice::hw\_ileds\_old, and usbdevice::ileds.

Referenced by \_setupusb().

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

Here is the call graph for this function:



Here is the caller graph for this function:



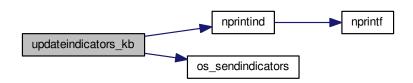
## 8.21.2.13 void updateindicators\_kb ( usbdevice \* kb, int force )

Definition at line 142 of file input.c.

References usbdevice::active, usbprofile::currentmode, DELAY\_SHORT, usbdevice::hw\_ileds, usbdevice::hw\_ileds\_old, I\_CAPS, I\_NUM, I\_SCROLL, usbdevice::ileds, usbmode::iontify, usbmode::ioff, usbmode::ion, nprintind(), os\_sendindicators(), OUTFIFO\_MAX, and usbdevice::profile.

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

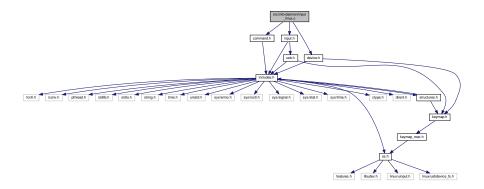
Here is the call graph for this function:



# 8.22 src/ckb-daemon/input\_linux.c File Reference

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

Include dependency graph for input\_linux.c:



#### **Functions**

- int uinputopen (struct uinput\_user\_dev \*indev, int mouse)
- int os inputopen (usbdevice \*kb)
- void os\_inputclose (usbdevice \*kb)
- static void isync (usbdevice \*kb)
- void os\_keypress (usbdevice \*kb, int scancode, int down)
- void os\_mousemove (usbdevice \*kb, int x, int y)
- void \* \_ledthread (void \*ctx)
- int os\_setupindicators (usbdevice \*kb)

# 8.22.1 Function Documentation

```
8.22.1.1 void* _ledthread ( void * ctx )
```

Definition at line 159 of file input\_linux.c.

References dmutex, usbdevice::hw ileds, usbdevice::uinput kb, and usbdevice::vtable.

Referenced by os\_setupindicators().

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

Here is the caller graph for this function:



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

Definition at line 101 of file input\_linux.c.

References ckb warn, usbdevice::uinput kb, and usbdevice::uinput mouse.

Referenced by os\_keypress(), and os\_mousemove().

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

Here is the caller graph for this function:

```
TOTAL STORM TOTAL
```

## 8.22.1.3 void os\_inputclose ( usbdevice \* kb )

Definition at line 70 of file input\_linux.c.

References ckb\_warn, usbdevice::uinput\_kb, and usbdevice::uinput\_mouse.

Referenced by closeusb().

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

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

Here is the caller graph for this function:



### 8.22.1.4 int os\_inputopen ( usbdevice \* kb )

Definition at line 49 of file input\_linux.c.

References usbdevice::fwversion, INDEX\_OF, keyboard, usbdevice::name, usbdevice::product, usbdevice::uinput\_kb, usbdevice::uinput\_mouse, uinputopen(), and usbdevice::vendor.

Referenced by \_setupusb().

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

Here is the call graph for this function:



Here is the caller graph for this function:



8.22.1.5 void os\_keypress ( usbdevice \* kb, int scancode, int down )

Definition at line 112 of file input\_linux.c.

References BTN\_WHEELDOWN, BTN\_WHEELUP, ckb\_warn, isync(), SCAN\_MOUSE, usbdevice::uinput\_kb, and usbdevice::uinput\_mouse.

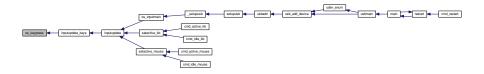
Referenced by inputupdate\_keys().

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

Here is the call graph for this function:



Here is the caller graph for this function:



8.22.1.6 void os\_mousemove ( usbdevice \*kb, int x, int y )

Definition at line 137 of file input linux.c.

References ckb warn, isync(), and usbdevice::uinput mouse.

Referenced by inputupdate(), and inputupdate\_keys().

137 {

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

Here is the call graph for this function:



Here is the caller graph for this function:



# 8.22.1.7 int os\_setupindicators ( usbdevice \* kb )

Definition at line 183 of file input\_linux.c.

References \_ledthread(), usbdevice::hw\_ileds, usbdevice::hw\_ileds\_old, and usbdevice::ileds.

Referenced by \_setupusb().

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

Here is the call graph for this function:



Here is the caller graph for this function:



## 8.22.1.8 int uinputopen ( struct uinput\_user\_dev \* indev, int mouse )

Definition at line 9 of file input\_linux.c.

References ckb\_err, and ckb\_warn.

Referenced by os inputopen().

```
10
         int fd = open("/dev/uinput", O_RDWR);
11
         if(fd < 0){</pre>
               // If that didn't work, try /dev/input/uinput instead
fd = open("/dev/input/uinput", O_RDWR);
if(fd < 0){</pre>
12
13
14
15
                    ckb_err("Failed to open uinput: %s\n", strerror(errno));
17
18
         // Enable all keys and mouse buttons
ioctl(fd, UI_SET_EVBIT, EV_KEY);
for(int i = 0; i < KEY_CNT; i++)</pre>
19
20
22
               ioctl(fd, UI_SET_KEYBIT, i);
         if (mouse) {
24
               // Enable mouse axes
               ioctl(fd, UI_SET_EVBIT, EV_REL);
for(int i = 0; i < REL_CNT; i++)</pre>
2.5
26
                    ioctl(fd, UI_SET_RELBIT, i);
27
         } else {
29
               // Enable LEDs
30
               ioctl(fd, UI_SET_EVBIT, EV_LED);
               for(int i = 0; i < LED_CNT; i++)
    ioctl(fd, UI_SET_LEDBIT, i);
// Eanble autorepeat</pre>
31
32
33
               ioctl(fd, UI_SET_EVBIT, EV_REP);
34
36
         // Enable sychronization
37
         ioctl(fd, UI_SET_EVBIT, EV_SYN);
         // Create the device
if(write(fd, indev, sizeof(*indev)) <= 0)</pre>
38
39
               ckb_warn("uinput write failed: %s\n", strerror(errno));
         if(ioctl(fd, UI_DEV_CREATE)){
    ckb_err("Failed to create uinput device: %s\n", strerror(errno));
41
43
               close(fd);
44
               return 0;
45
46
         return fd + 1;
```

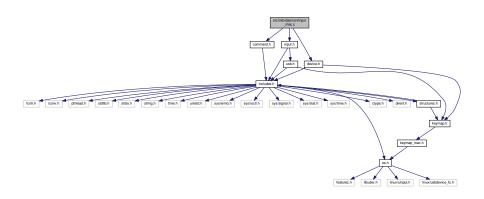
Here is the caller graph for this function:



# 8.23 src/ckb-daemon/input\_mac.c File Reference

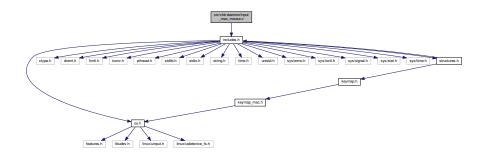
```
#include "command.h"
#include "device.h"
#include "input.h"
```

Include dependency graph for input\_mac.c:



# 8.24 src/ckb-daemon/input\_mac\_mouse.c File Reference

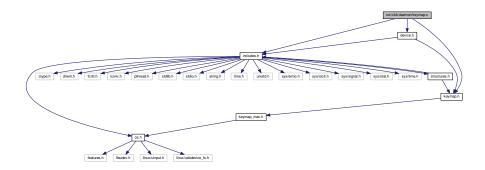
#include "includes.h"
Include dependency graph for input\_mac\_mouse.c:



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

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

Include dependency graph for keymap.c:



## **Macros**

• #define BUTTON\_HID\_COUNT 5

### **Functions**

- void hid\_kb\_translate (unsigned char \*kbinput, int endpoint, int length, const unsigned char \*urbinput)
- void hid\_mouse\_translate (unsigned char \*kbinput, short \*xaxis, short \*yaxis, int endpoint, int length, const unsigned char \*urbinput)
- void corsair kbcopy (unsigned char \*kbinput, int endpoint, const unsigned char \*urbinput)
- · void corsair\_mousecopy (unsigned char \*kbinput, int endpoint, const unsigned char \*urbinput)

### **Variables**

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

# 8.25.1 Macro Definition Documentation

# 8.25.1.1 #define BUTTON\_HID\_COUNT 5

Definition at line 364 of file keymap.c.

Referenced by corsair\_mousecopy(), and hid\_mouse\_translate().

## 8.25.2 Function Documentation

8.25.2.1 void corsair\_kbcopy ( unsigned char \* kbinput, int endpoint, const unsigned char \* urbinput )

Definition at line 394 of file keymap.c.

References N\_KEYBYTES\_HW.

Referenced by os\_inputmain().

Here is the caller graph for this function:



8.25.2.2 void corsair\_mousecopy ( unsigned char \* kbinput, int endpoint, const unsigned char \* urbinput )

Definition at line 403 of file keymap.c.

References BUTTON\_HID\_COUNT, CLEAR\_KEYBIT, MOUSE\_BUTTON\_FIRST, N\_BUTTONS\_HW, and SET\_K-EYBIT.

Referenced by os\_inputmain().

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

Here is the caller graph for this function:



8.25.2.3 void hid\_kb\_translate ( unsigned char \* kbinput, int endpoint, int length, const unsigned char \* urbinput )

Definition at line 223 of file keymap.c.

References ckb\_warn, CLEAR\_KEYBIT, and SET\_KEYBIT.

Referenced by os\_inputmain().

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

```
120, 121, 122, 123, 124, 125, 126, 127, 128, 129, 136, 137, 138, 139, 140, 141,
             60, 48, 62, 61, 91, 90, 67, 68, 142, 143, 99, 101, -2, 130, 131, 97, -2, 133, 134, 135, -2, 96, -2, 132, -2, -2, 71, 71, 71, 71, -1, -1,
243
244
245
         switch(endpoint){
246
247
         case 1:
         case -1:
248
249
             // EP 1: 6KRO input (RGB and non-RGB)
              // Clear previous input
for(int i = 0; i < 256; i++){</pre>
250
2.51
                   if(hid_codes[i] >= 0)
252
                       CLEAR_KEYBIT(kbinput, hid_codes[i]);
253
254
              // Set new input
for(int i = 0; i < 8; i++){
255
256
                  if((urbinput[0] >> i) & 1)
    SET_KEYBIT(kbinput, hid_codes[i + 224]);
2.57
258
259
260
              for(int i = 2; i < length; i++) {</pre>
                   if(urbinput[i] > 3){
261
262
                       int scan = hid_codes[urbinput[i]];
                        if(scan >= 0)
263
                            SET_KEYBIT(kbinput, scan);
2.64
265
                       else
266
                            ckb_warn("Got unknown key press %d on EP 1\n", urbinput[i]);
267
                  }
268
269
             break;
270
         case -2:
             // EP 2 RGB: NKRO input
271
272
              if (urbinput[0] == 1) {
273
                  // Type 1: standard key
274
                   if(length != 21)
275
                        return;
                   for(int bit = 0; bit < 8; bit++) {
    if((urbinput[1] >> bit) & 1)
276
277
                            SET_KEYBIT(kbinput, hid_codes[bit + 224]);
278
279
280
                            CLEAR_KEYBIT(kbinput, hid_codes[bit + 224]);
281
282
                   for(int byte = 0; byte < 19; byte++) {</pre>
                       char input = urbinput[byte + 2];
for(int bit = 0; bit < 8; bit++){
  int keybit = byte * 8 + bit;</pre>
283
284
285
                            int scan = hid_codes[keybit];
286
287
                            if((input >> bit) & 1){
288
                                 if(scan >= 0)
                                     SET_KEYBIT(kbinput, hid_codes[keybit]);
289
290
                                 else
                            291
292
293
                                 CLEAR_KEYBIT(kbinput, hid_codes[keybit]);
294
                       }
295
296
                  break:
297
              } else if(urbinput[0] == 2)
298
                          // Type 2: media key (fall through)
                 ;
299
300
                  break; // No other known types
301
         case 2:
              // EP 2 Non-RGB: media keys
302
              CLEAR_KEYBIT(kbinput, 97);
CLEAR_KEYBIT(kbinput, 98);
                                                         // mute
303
304
                                                         // stop
              CLEAR_KEYBIT(kbinput, 99);
CLEAR_KEYBIT(kbinput, 100);
305
                                                         // prev
306
                                                         // play
307
              CLEAR_KEYBIT(kbinput, 101);
                                                         // next
              CLEAR_KEYBIT(kbinput, 130);
CLEAR_KEYBIT(kbinput, 131);
for(int i = 0; i < length; i++){</pre>
308
                                                         // volup
                                                         // voldn
309
310
                  switch(urbinput[i]){
311
312
                   case 181:
313
                       SET_KEYBIT(kbinput, 101);
                                                         // next
314
                      break;
315
                   case 182:
                      SET_KEYBIT(kbinput, 99);
break;
316
                                                         // prev
317
                   case 183:
318
319
                      SET_KEYBIT(kbinput, 98);
                                                         // stop
                       break;
320
                   case 205:
321
                      SET_KEYBIT(kbinput, 100);
322
                                                        // plav
323
                       break;
                   case 226:
324
325
                      SET_KEYBIT(kbinput, 97);
                                                         // mute
326
                       break;
327
                   case 233:
328
                       SET_KEYBIT(kbinput, 130);
                                                        // volup
```

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

Here is the caller graph for this function:



8.25.2.4 void hid\_mouse\_translate ( unsigned char \* kbinput, short \* xaxis, short \* yaxis, int endpoint, int length, const unsigned char \* urbinput )

Definition at line 366 of file keymap.c.

References BUTTON\_HID\_COUNT, CLEAR\_KEYBIT, MOUSE\_BUTTON\_FIRST, MOUSE\_EXTRA\_FIRST, and S-ET\_KEYBIT.

Referenced by os\_inputmain().

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

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

Here is the caller graph for this function:



#### 8.25.3 Variable Documentation

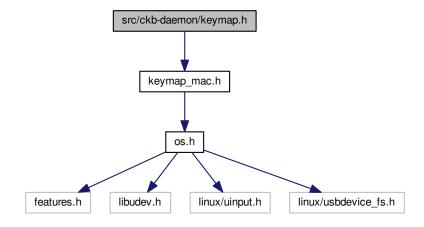
8.25.3.1 const key keymap[(((152+3+12)+25)+11)]

Definition at line 5 of file keymap.c.

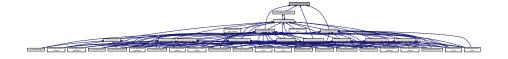
Referenced by \_cmd\_get(), \_cmd\_macro(), cmd\_bind(), cmd\_rebind(), cmd\_rgb(), initbind(), inputupdate\_keys(), nprintkey(), printrgb(), readcmd(), and setactive\_kb().

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

#include "keymap\_mac.h"
Include dependency graph for keymap.h:



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



#### **Data Structures**

· struct key

#### **Macros**

- #define KEY NONE -1
- #define KEY\_CORSAIR -2
- #define KEY UNBOUND -3
- #define BTN WHEELUP 0x1f01
- #define BTN WHEELDOWN 0x1f02
- #define KEY\_BACKSLASH\_ISO KEY\_BACKSLASH
- #define N KEYS HW 152
- #define N\_KEYBYTES\_HW ((N\_KEYS\_HW + 7) / 8)
- #define N\_KEY\_ZONES 3
- #define N\_KEYS\_EXTRA 12
- #define N BUTTONS HW 20
- #define N\_BUTTONS\_EXTENDED 25
- #define MOUSE\_BUTTON\_FIRST (N\_KEYS\_HW + N\_KEY\_ZONES + N\_KEYS\_EXTRA)
- #define MOUSE\_EXTRA\_FIRST (MOUSE\_BUTTON\_FIRST + N\_BUTTONS\_HW)
- #define N\_KEYS\_INPUT (MOUSE\_BUTTON\_FIRST + N\_BUTTONS\_EXTENDED)
- #define N\_KEYBYTES\_INPUT ((N\_KEYS\_INPUT + 7) / 8)
- #define LED MOUSE N KEYS HW
- #define N\_MOUSE\_ZONES 5
- #define N\_MOUSE\_ZONES\_EXTENDED 11
- #define LED\_DPI (LED\_MOUSE + 2)
- #define N KEYS EXTENDED (N KEYS INPUT + N MOUSE ZONES EXTENDED)
- #define N\_KEYBYTES\_EXTENDED ((N\_KEYS\_EXTENDED + 7) / 8)
- #define SCAN\_SILENT 0x8000
- #define SCAN KBD 0
- #define SCAN MOUSE 0x1000

# **Functions**

- void hid\_kb\_translate (unsigned char \*kbinput, int endpoint, int length, const unsigned char \*urbinput)
- void hid\_mouse\_translate (unsigned char \*kbinput, short \*xaxis, short \*yaxis, int endpoint, int length, const unsigned char \*urbinput)
- void corsair\_kbcopy (unsigned char \*kbinput, int endpoint, const unsigned char \*urbinput)
- · void corsair\_mousecopy (unsigned char \*kbinput, int endpoint, const unsigned char \*urbinput)

# **Variables**

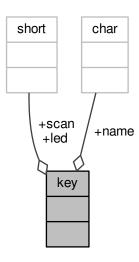
const key keymap [(((152+3+12)+25)+11)]

## 8.26.1 Data Structure Documentation

8.26.1.1 struct key

Definition at line 49 of file keymap.h.

Collaboration diagram for key:



# **Data Fields**

short	led	
const char *	name	
short	scan	

# 8.26.2 Macro Definition Documentation

8.26.2.1 #define BTN\_WHEELDOWN 0x1f02

Definition at line 13 of file keymap.h.

Referenced by os\_keypress().

8.26.2.2 #define BTN\_WHEELUP 0x1f01

Definition at line 12 of file keymap.h.

Referenced by os\_keypress().

8.26.2.3 #define KEY\_BACKSLASH\_ISO KEY\_BACKSLASH

Definition at line 20 of file keymap.h.

8.26.2.4 #define KEY\_CORSAIR -2

Definition at line 8 of file keymap.h.

8.26.2.5 #define KEY\_NONE -1

Definition at line 7 of file keymap.h.

8.26.2.6 #define KEY\_UNBOUND -3

Definition at line 9 of file keymap.h.

Referenced by cmd\_unbind().

8.26.2.7 #define LED\_DPI (LED\_MOUSE + 2)

Definition at line 43 of file keymap.h.

Referenced by loadrgb\_mouse(), and savergb\_mouse().

8.26.2.8 #define LED\_MOUSE N\_KEYS\_HW

Definition at line 39 of file keymap.h.

Referenced by isblack(), loaddpi(), loadrgb\_mouse(), rgbcmp(), savedpi(), savergb\_mouse(), and updatergb\_mouse().

8.26.2.9 #define MOUSE\_BUTTON\_FIRST (N\_KEYS\_HW + N\_KEY\_ZONES + N\_KEYS\_EXTRA)

Definition at line 33 of file keymap.h.

Referenced by corsair\_mousecopy(), and hid\_mouse\_translate().

8.26.2.10 #define MOUSE\_EXTRA\_FIRST (MOUSE\_BUTTON\_FIRST + N\_BUTTONS\_HW)

Definition at line 34 of file keymap.h.

Referenced by hid\_mouse\_translate().

8.26.2.11 #define N\_BUTTONS\_EXTENDED 25

Definition at line 32 of file keymap.h.

8.26.2.12 #define N\_BUTTONS\_HW 20

Definition at line 31 of file keymap.h.

Referenced by corsair\_mousecopy().

8.26.2.13 #define N\_KEY\_ZONES 3

Definition at line 27 of file keymap.h.

8.26.2.14 #define N\_KEYBYTES\_EXTENDED ((N\_KEYS\_EXTENDED + 7) / 8)

Definition at line 46 of file keymap.h.

8.26.2.15 #define N\_KEYBYTES\_HW ((N\_KEYS\_HW + 7) / 8)

Definition at line 25 of file keymap.h.

Referenced by corsair kbcopy().

8.26.2.16 #define N\_KEYBYTES\_INPUT ((N\_KEYS\_INPUT + 7) / 8)

Definition at line 37 of file keymap.h.

Referenced by \_cmd\_macro(), inputupdate\_keys(), and macromask().

8.26.2.17 #define N\_KEYS\_EXTENDED (N\_KEYS\_INPUT + N\_MOUSE\_ZONES\_EXTENDED)

Definition at line 45 of file keymap.h.

Referenced by printrgb(), and readcmd().

8.26.2.18 #define N\_KEYS\_EXTRA 12

Definition at line 29 of file keymap.h.

8.26.2.19 #define N\_KEYS\_HW 152

Definition at line 24 of file keymap.h.

Referenced by loadrgb\_kb(), makergb\_512(), rgbcmp(), and setactive\_kb().

8.26.2.20 #define N\_KEYS\_INPUT (MOUSE\_BUTTON\_FIRST + N\_BUTTONS\_EXTENDED)

Definition at line 36 of file keymap.h.

 $Referenced \ by \ \_cmd\_get(), \ \_cmd\_macro(), \ cmd\_bind(), \ cmd\_notify(), \ cmd\_rebind(), \ cmd\_unbind(), \ initbind(), \ and \ input update\_keys().$ 

8.26.2.21 #define N\_MOUSE\_ZONES 5

Definition at line 40 of file keymap.h.

Referenced by isblack(), loaddpi(), rgbcmp(), savedpi(), and updatergb\_mouse().

8.26.2.22 #define N\_MOUSE\_ZONES\_EXTENDED 11

Definition at line 41 of file keymap.h.

8.26.2.23 #define SCAN\_KBD 0

Definition at line 57 of file keymap.h.

8.26.2.24 #define SCAN\_MOUSE 0x1000

Definition at line 58 of file keymap.h.

Referenced by os\_keypress().

## 8.26.2.25 #define SCAN\_SILENT 0x8000

Definition at line 56 of file keymap.h.

Referenced by inputupdate\_keys().

# 8.26.3 Function Documentation

8.26.3.1 void corsair\_kbcopy ( unsigned char \* kbinput, int endpoint, const unsigned char \* urbinput )

Definition at line 394 of file keymap.c.

References N\_KEYBYTES\_HW.

Referenced by os\_inputmain().

Here is the caller graph for this function:

```
cosair Moopy cs_inputmain setupuab setupuab usbadd usb_add_device usbmain main setupuab cnd_restart
```

8.26.3.2 void corsair\_mousecopy ( unsigned char \* kbinput, int endpoint, const unsigned char \* urbinput )

Definition at line 403 of file keymap.c.

References BUTTON\_HID\_COUNT, CLEAR\_KEYBIT, MOUSE\_BUTTON\_FIRST, N\_BUTTONS\_HW, and SET\_K-EYBIT.

Referenced by os\_inputmain().

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

Here is the caller graph for this function:



8.26.3.3 void hid\_kb\_translate ( unsigned char \* kbinput, int endpoint, int length, const unsigned char \* urbinput )

Definition at line 223 of file keymap.c.

References ckb warn, CLEAR KEYBIT, and SET KEYBIT.

Referenced by os inputmain().

```
223
224
         if(length < 1)</pre>
225
              return;
         // LUT for HID -> Corsair scancodes (-1 for no scan code, -2 for currently unsupported)
226
227
         // Modified from Linux drivers/hid/usbhid/usbkbd.c, key codes replaced with array indices and K95 keys
        added
228
         static const short hid_codes[256]
              -1,
                               -1,
229
                   -1,
                         -1,
                                     37,
                                           54,
                                                 52,
                                                       39,
                                                            27,
                                                                  40,
                                                                        41,
                                                                              42,
                                                                                    32,
                                                                                          43,
                                                                                                44,
                                                                                                     45,
                                                            31,
                                                                        26,
                                                                              51,
230
              56,
                   55,
                         33,
                               34,
                                     25,
                                           28,
                                                 38,
                                                       29,
                                                                                    30,
                                                                                          50,
                                                                                                13,
                                                                  53,
                                                                                                      14,
                                           20,
                                                            82,
                                                                                                84,
231
              15.
                   16,
                         17,
                               18,
                                     19,
                                                 21,
                                                      22,
                                                                   0,
                                                                        86,
                                                                              24,
                                                                                    64,
                                                                                          23,
                                                                                                      35,
                                                            59,
                                                                                                       6,
232
              79, 80,
                        81,
                               46,
                                     47,
                                           12,
                                                 57,
                                                       58,
                                                                  36,
                                                                         1,
                                                                               2,
                                                                                     3,
                                                                                           4.
                                                                                                 5.
                                                             75,
                    8,
                                                                               78,
233
               7.
                          9,
                               10,
                                     11,
                                           72,
                                                 73,
                                                       74.
                                                                   76,
                                                                                    87,
                                                                                          88.
                                                                                                89.
                                                                                                      95.
234
              93,
                   94,
                         92, 102, 103, 104, 105, 106,
                                                           107, 115, 116, 117, 112, 113,
                                                                                              114, 108,
235
             109, 110, 118, 119, 49,
                                          69,
                                                 -2,
                                                      -2,
                                                                        -2,
                                                                              -2,
                                                                                                -2,
                                                                                                      -2,
                         -2,
                                           -2,
                                                             98,
                                                                        -2,
                                                                                          -1,
237
             130, 131,
                         -1,
                               -1,
                                     -1,
                                           -2,
                                                 -1,
                                                      -2,
                                                             -2,
                                                                  -2,
                                                                              -2,
                                                                                    -2,
                                                                                                -1,
                                                                                                      -1.
                               -2,
                                          -1,
                                                 -1,
                                                      -1,
                                                            -1,
                                                                  -1,
                                                                              -1,
                                                                                    -1,
                                                                                          -1,
                                                                                                     -1,
238
             -2, -2,
                         -2,
                                     -2,
                                                                        -1,
                                                                                                -1,
239
             -1, -1,
                         -1,
                               -1.
                                     -1.
                                           -1,
                                                 -1,
                                                      -1,
                                                            -1,
                                                                  -1,
                                                                        -1,
                                                                              -1.
                                                                                    -1.
                                                                                          -1,
                                                                                               -1.
                                                                                                     -1.
             -1, -1,
-1, -1,
                         -1.
                                                                   -1.
240
                               -1.
                                     -1.
                                           -1.
                                                 -1.
                                                                        -1.
                                                                                    -1.
                                                                                          -1.
241
                         -1,
                               -1,
                                     -1,
                                           -1,
                                                 -1,
                                                                        -1,
                                                                                    -3,
                                                                                                -1,
                                                                                                      -1,
                                                                                                           // < - -3 = non-RGB
       program key
242
             120, 121, 122, 123, 124, 125, 126, 127, 128, 129, 136, 137, 138, 139, 140, 141,
                   48,
             60, 48, 62, 61, 91, -2, 133, 134, 135, -2,
                                           90, 67, 68, 142, 143, 96, -2, 132, -2, -2,
243
                                                                        99, 101,
                                                                                    -2, 130, 131,
                                                                                    71.
244
                                                                        71.
                                                                              71.
                                                                                          71.
245
246
         switch(endpoint){
247
         case 1:
248
         case -1:
249
              // EP 1: 6KRO input (RGB and non-RGB)
              // Clear previous input
for(int i = 0; i < 256; i++) {</pre>
250
2.51
                  if(hid_codes[i] >= 0)
252
253
                       CLEAR_KEYBIT(kbinput, hid_codes[i]);
254
              // Set new input
for(int i = 0; i < 8; i++) {
    if((urbinput[0] >> i) & 1)
255
256
257
258
                       SET_KEYBIT(kbinput, hid_codes[i + 224]);
259
260
              for(int i = 2; i < length; i++) {</pre>
261
                  if(urbinput[i] > 3){
                       int scan = hid_codes[urbinput[i]];
if(scan >= 0)
2.62
263
                           SET_KEYBIT(kbinput, scan);
264
265
                            ckb_warn("Got unknown key press %d on EP 1\n", urbinput[i]);
267
268
269
             break:
270
         case -2:
271
             // EP 2 RGB: NKRO input
272
              if (urbinput[0] == 1) {
273
                  // Type 1: standard key
274
                  if(length != 21)
275
                       return;
276
                  for(int bit = 0; bit < 8; bit++){</pre>
                       if((urbinput[1] >> bit) & 1)
277
                           SET_KEYBIT(kbinput, hid_codes[bit + 224]);
279
280
                            CLEAR_KEYBIT(kbinput, hid_codes[bit + 224]);
281
                  for(int byte = 0; byte < 19; byte++) {</pre>
282
                       char input = urbinput[byte + 2];
for(int bit = 0; bit < 8; bit++){
   int keybit = byte * 8 + bit;</pre>
283
284
285
                            int scan = hid_codes[keybit];
if((input >> bit) & 1){
286
287
288
                                 if(scan >= 0)
                                     SET_KEYBIT(kbinput, hid_codes[keybit]);
289
290
291
                                    ckb_warn("Got unknown key press %d on EP 2\n", keybit);
292
293
                                CLEAR_KEYBIT(kbinput, hid_codes[keybit]);
294
295
296
                  break;
297
              } else if(urbinput[0] == 2)
```

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

for(int bit = 0; bit < 8; bit++){
347
348
349
                     int keybit = byte * 8 + bit;
350
                      int scan = hid_codes[keybit];
                      if((input >> bit) & 1) {
    if(scan >= 0)
351
352
                              SET_KEYBIT(kbinput, hid_codes[keybit]);
353
354
355
                              ckb_warn("Got unknown key press %d on EP 3\n", keybit);
356
                      } else if(scan >= 0)
357
                          CLEAR_KEYBIT(kbinput, hid_codes[keybit]);
358
                 }
359
360
             break;
362 }
```

Here is the caller graph for this function:



8.26.3.4 void hid\_mouse\_translate ( unsigned char \* kbinput, short \* xaxis, short \* yaxis, int endpoint, int length, const unsigned char \* urbinput )

Definition at line 366 of file keymap.c.

References BUTTON\_HID\_COUNT, CLEAR\_KEYBIT, MOUSE\_BUTTON\_FIRST, MOUSE\_EXTRA\_FIRST, and S-ET\_KEYBIT.

Referenced by os\_inputmain().

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

Here is the caller graph for this function:



# 8.26.4 Variable Documentation

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

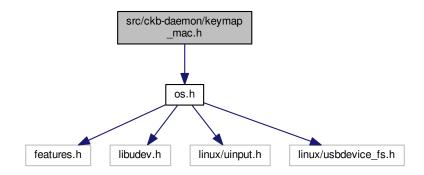
Definition at line 5 of file keymap.c.

Referenced by \_cmd\_get(), \_cmd\_macro(), cmd\_bind(), cmd\_rebind(), cmd\_rgb(), initbind(), inputupdate\_keys(), nprintkey(), printrgb(), readcmd(), and setactive\_kb().

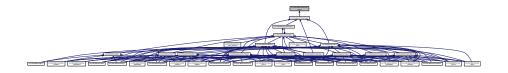
# 8.27 src/ckb-daemon/keymap\_mac.h File Reference

#include "os.h"

Include dependency graph for keymap\_mac.h:



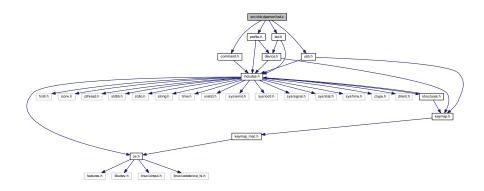
This graph shows which files directly or indirectly include this file:



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

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

Include dependency graph for led.c:



# **Functions**

- void cmd\_rgb (usbdevice \*kb, usbmode \*mode, int dummy, int keyindex, const char \*code)
- static uchar iselect (const char \*led)
- void cmd\_ioff (usbdevice \*kb, usbmode \*mode, int dummy1, int dummy2, const char \*led)

- void cmd\_ion (usbdevice \*kb, usbmode \*mode, int dummy1, int dummy2, const char \*led)
- void cmd\_iauto (usbdevice \*kb, usbmode \*mode, int dummy1, int dummy2, const char \*led)
- void cmd\_inotify (usbdevice \*kb, usbmode \*mode, int nnumber, int dummy, const char \*led)
- static int has\_key (const char \*name, const usbdevice \*kb)
- char \* printrgb (const lighting \*light, const usbdevice \*kb)

#### 8.28.1 Function Documentation

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

Definition at line 54 of file led.c.

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

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

Here is the call graph for this function:



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

Definition at line 62 of file led.c.

References usbmode::inotify, and iselect().

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

Here is the call graph for this function:



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

Definition at line 38 of file led.c.

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

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

Here is the call graph for this function:



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

Definition at line 46 of file led.c.

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

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

Here is the call graph for this function:



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

Definition at line 6 of file led.c.

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

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

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

8.28.1.6 static int has\_key ( const char \* name, const usbdevice \* kb ) [static]

Definition at line 73 of file led.c.

References IS\_K65, IS\_K95, IS\_MOUSE, IS\_SABRE, IS\_SCIMITAR, usbdevice::product, and usbdevice::vendor. Referenced by printrgb().

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

Here is the caller graph for this function:



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

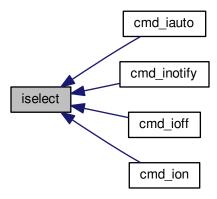
Definition at line 25 of file led.c.

References I\_CAPS, I\_NUM, and I\_SCROLL.

Referenced by cmd\_iauto(), cmd\_inotify(), cmd\_ioff(), and cmd\_ion().

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

Here is the caller graph for this function:



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

Definition at line 102 of file led.c.

References lighting::b, lighting::g, has\_key(), keymap, key::led, N\_KEYS\_EXTENDED, key::name, and lighting::r.

```
Referenced by _cmd_get().
```

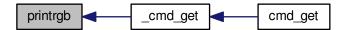
```
102
103
          uchar r[N_KEYS_EXTENDED], q[N_KEYS_EXTENDED], b[
        N_KEYS_EXTENDED];
104
          const uchar* mr = light->r;
105
          const uchar* mg = light->g;
          const uchar* mg = light->b;
const uchar* mb = light->b;
for(int i = 0; i < N_KEYS_EXTENDED; i++){
    // Translate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

```
123
                 strncpy(names[i], name, 11);
124
         // Check to make sure these aren't all the same color
125
        int same = 1;
for(int i = 1; i < N_KEYS_EXTENDED; i++) {</pre>
126
127
             if(!names[i][0])
128
129
                  continue;
130
             if(r[i] != r[0] || g[i] != g[0] || b[i] != b[0]){
                 same = 0;
131
132
                  break;
133
134
135
         // If they are, just output that color
136
        if(same){
137
             char* buffer = malloc(7);
             snprintf(buffer, 7, "%02x%02x%02x", r[0], g[0], b[0]);
138
139
             return buffer:
140
141
        const int BUFFER_LEN = 4096;
                                            // Should be more than enough to fit all keys
142
        char* buffer = malloc(BUFFER_LEN);
         int length = 0;
for(int i = 0; i < N_KEYS_EXTENDED; i++){</pre>
143
144
             if(!names[i][0])
145
146
                  continue;
147
             // Print the key name
148
             int newlen = 0;
149
             snprintf(buffer + length, BUFFER\_LEN - length, length == 0 ? "\$s\$n" : " \$s\$n", names[i], \&newlen);
150
             length += newlen;
             // Look ahead to see if any other keys have this color. If so, print them here as well. uchar kr=r[i], kg=g[i], kb=b[i]; for (int j=i+1; j < N_KEYS_EXTENDED; j++) (
151
152
153
154
                  if(!names[j][0])
155
156
                 if(r[j] != kr || g[j] != kg || b[j] != kb)
157
                  snprintf(buffer + length, BUFFER_LEN - length, ",%s%n", names[j], &newlen);
158
159
                  length += newlen;
                  // Erase the key's name so it won't get printed later
160
161
                  names[j][0] = 0;
162
             // Print the color snprintf(buffer + length, BUFFER_LEN - length, ":%02x%02x%01, kr, kg, kb, &newlen);
163
164
165
             length += newlen;
166
167
         return buffer;
168 }
```

Here is the call graph for this function:



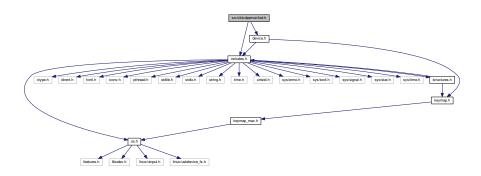
Here is the caller graph for this function:



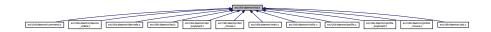
## 8.29 src/ckb-daemon/led.h File Reference

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

Include dependency graph for led.h:



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



#### **Functions**

- int updatergb\_kb (usbdevice \*kb, int force)
- int updatergb\_mouse (usbdevice \*kb, int force)
- int savergb\_kb (usbdevice \*kb, lighting \*light, int mode)
- int savergb\_mouse (usbdevice \*kb, lighting \*light, int mode)
- int loadrgb\_kb (usbdevice \*kb, lighting \*light, int mode)
- int loadrgb mouse (usbdevice \*kb, lighting \*light, int mode)
- char \* printrgb (const lighting \*light, const usbdevice \*kb)
- void cmd\_rgb (usbdevice \*kb, usbmode \*mode, int dummy, int keyindex, const char \*code)
- void cmd\_ioff (usbdevice \*kb, usbmode \*mode, int dummy1, int dummy2, const char \*led)
- void cmd\_ion (usbdevice \*kb, usbmode \*mode, int dummy1, int dummy2, const char \*led)
- void cmd\_iauto (usbdevice \*kb, usbmode \*mode, int dummy1, int dummy2, const char \*led)
- void cmd\_inotify (usbdevice \*kb, usbmode \*mode, int nnumber, int dummy, const char \*led)

# 8.29.1 Function Documentation

8.29.1.1 void cmd\_iauto ( usbdevice \* kb, usbmode \* mode, int dummy1, int dummy2, const char \* led )

Definition at line 54 of file led.c.

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

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

Here is the call graph for this function:



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

Definition at line 62 of file led.c.

References usbmode::inotify, and iselect().

Here is the call graph for this function:



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

Definition at line 38 of file led.c.

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

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

Here is the call graph for this function:



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

Definition at line 46 of file led.c.

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

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

Here is the call graph for this function:



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

Definition at line 6 of file led.c.

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

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

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

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

Definition at line 181 of file led keyboard.c.

References lighting::b, ckb\_err, usbdevice::fwversion, lighting::g, MSG\_SIZE, N\_KEYS\_HW, lighting::r, usbrecv, and usbsend.

Referenced by hwloadmode().

```
181
        if(kb->fwversion >= 0x0120){
182
183
            uchar data_pkt[12][MSG_SIZE] = {
184
                 { 0x0e, 0x14, 0x03, 0x01, 0x01, mode + 1, 0x01 },
185
                   0xff, 0x01, 60, 0 },
186
                   0xff, 0x02, 60, 0 },
187
                 { 0xff, 0x03, 24, 0 }
                 { 0x0e, 0x14, 0x03, 0x01, 0x01, mode + 1, 0x02 },
188
189
                 { 0xff, 0x01, 60, 0 },
                   0xff, 0x02, 60, 0
191
                   0xff, 0x03, 24, 0 },
192
                  0x0e, 0x14, 0x03, 0x01, 0x01, mode + 1, 0x03 },
193
                   0xff, 0x01, 60, 0 },
194
                 { 0xff, 0x02, 60, 0 },
195
                 { 0xff, 0x03, 24, 0 },
196
197
            uchar in_pkt[4][MSG_SIZE] = {
198
                 { 0x0e, 0x14, 0x03, 0x01 },
                 { 0xff, 0x01, 60, 0 },
199
200
                 { 0xff, 0x02, 60, 0 },
                 { 0xff, 0x03, 24, 0 },
201
202
203
207
208
            uchar cmp_pkt[4][4] = {
                 { 0x0e, 0x14, 0x03, 0x01 },
{ 0x0e, 0xff, 0x01, 60 },
209
210
211
                   0x0e, 0xff, 0x02, 60 },
                 { 0x0e, 0xff, 0x03, 24 },
213
            // Read colors
214
            uchar* colors[3] = { light->r, light->g, light->b };
215
            for (int clr = 0; clr < 3; clr++) {</pre>
216
                 for (int i = 0; i < 4; i++) {
                     if(!usbrecv(kb, data_pkt[i + clr * 4], in_pkt[i]))
218
                          ceturn -1;
219
220
                     // Make sure the first four bytes match
221
                     // see comment above
                     // if(memcmp(p, data_pkt[i + clr * 4], 4)){
222
                     if (memcmp(in_pkt[i], (kb->fwversion >= 0x0205)? cmp_pkt[i] : data_pkt[i + clr * 4
223
      ], 4)) {
224
                         ckb_err("Bad input header\n");
225
                         ckb_err("color = %d, i = %d, mode = %d\nInput(Antwort): %2.2x %2.2x %2.2x %2.2x
       $2.2x $2.2x $2.2x $2.2x\nOutput (Frage): $2.2x $2.2x $2.2x $2.2x\n", clr, i, mode,
226
                              in_pkt[i][0], in_pkt[i][1], in_pkt[i][2], in_pkt[i][3], in_pkt[i][4], in_pkt[i][5],
       in_pkt[i][6], in_pkt[i][7],
227
                                  // data_pkt[i + clr * 4][0],
                                                                  data_pkt[i + clr * 4 ][1], data_pkt[i + clr *
       4 ][2], data_pkt[i + clr * 4 ][3]);
228
                              cmp_pkt[i][0], cmp_pkt[i][1], cmp_pkt[i][2], cmp_pkt[i][3]);
229
                         in_pkt[2][0] = 0x99;
in_pkt[2][1] = 0x99;
230
231
                         in_pkt[2][2] = 0x99;
232
                         in_pkt[2][3] = 0x99;
233
                         usbrecv(kb, in_pkt[2], in_pkt[2]); // just to find it in the wireshark log
234
                          return -1;
235
                     }
236
                 // Copy colors to lighting. in_pkt[0] is irrelevant.
237
                 memcpy(colors[clr], in_pkt[1] + 4, 60);
memcpy(colors[clr] + 60, in_pkt[2] + 4, 60);
238
239
240
                 memcpy(colors[clr] + 120, in_pkt[3] + 4, 24);
241
            }
242
        } else {
            uchar data_pkt[5][MSG_SIZE] = {
243
244
                 { 0x0e, 0x14, 0x02, 0x01, 0x01, mode + 1, 0 },
                   0xff, 0x01, 60, 0 },
245
                   0xff, 0x02, 60, 0 },
246
247
                  0xff, 0x03, 60, 0 },
2.48
                 { 0xff, 0x04, 36, 0 },
249
            };
250
            uchar in_pkt[4][MSG_SIZE] =
251
                 { 0xff, 0x01, 60, 0 },
```

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

g = 7 - (mg[i_2] >> 4);
283
284
                         b = 7 - (mb[i_2] >> 4);
285
                    } else {
    r = 7 - (mr[i_2] & 0x0F);
    g = 7 - (mg[i_2] & 0x0F);
286
287
288
289
                         b = 7 - (mb[i_2] \& 0x0F);
290
                     ^{\prime}// Scale 3-bit values up to 8 bits.
291
                    light->r[i] = r << 5 | r << 2 | r >> 1;
light->g[i] = g << 5 | g << 2 | g >> 1;
292
293
                    light->b[i] = b << 5 | b << 2 | b >> 1;
294
295
296
297
          return 0;
298 }
```

Here is the caller graph for this function:



8.29.1.7 int loadrgb\_mouse ( usbdevice \* kb, lighting \* light, int mode )

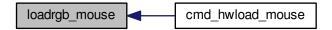
Definition at line 81 of file led\_mouse.c.

References lighting::b, ckb\_err, lighting::g, IS\_SABRE, IS\_SCIMITAR, LED\_DPI, LED\_MOUSE, MSG\_SIZE, lighting::r, and usbrecv.

Referenced by cmd\_hwload\_mouse().

```
86
        for (int i = 0; i < zonecount; i++) {
             if(!usbrecv(kb, data_pkt, in_pkt))
88
                   return -1;
              if (memcmp(in_pkt, data_pkt, 4)) {
    ckb_err("Bad input header\n");
89
90
                   return -2;
91
92
93
              // Copy data
              int led = LED_MOUSE + i;
if(led >= LED_DPI)
94
9.5
                   led++;
                                        // Skip DPI light
96
              light->r[led] = in_pkt[4];
97
              light->g[led] = in_pkt[5];
light->b[led] = in_pkt[6];
98
99
100
               // Set packet for next zone
101
               data_pkt[2]++;
102
103
          return 0;
104 }
```

Here is the caller graph for this function:



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

Definition at line 102 of file led.c.

References lighting::b, lighting::g, has\_key(), keymap, key::led, N\_KEYS\_EXTENDED, key::name, and lighting::r. Referenced by \_cmd\_get().

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

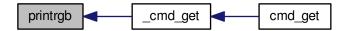
```
135
        // If they are, just output that color
136
        if(same) {
137
             char* buffer = malloc(7);
             snprintf(buffer, 7, "%02x%02x%02x", r[0], g[0], b[0]);
138
139
             return buffer;
140
        const int BUFFER_LEN = 4096;
                                           // Should be more than enough to fit all keys
141
142
        char* buffer = malloc(BUFFER_LEN);
        int length = 0;
for(int i = 0; i < N_KEYS_EXTENDED; i++) {
   if(!names[i][0])</pre>
143
144
145
146
                 continue;
             // Print the key name
147
148
             int newlen = 0;
149
             snprintf(buffer + length, BUFFER\_LEN - length, length == 0 ? "\$s\$n" : " \$s\$n", names[i], \&newlen);
150
             length += newlen;
             // Look ahead to see if any other keys have this color. If so, print them here as well.
151
             uchar kr = r[i], kg = g[i], kb = b[i];

for(int j = i + 1; j < N_KEYS_EXTENDED; j++) {
152
153
154
                 if(!names[j][0])
155
156
                 if(r[j] != kr || g[j] != kg || b[j] != kb)
157
                 snprintf(buffer + length, BUFFER_LEN - length, ",%s%n", names[j], &newlen);
158
159
                 length += newlen;
                 // Erase the key's name so it won't get printed later
160
161
                 names[j][0] = 0;
162
             // Print the color
163
             snprintf(buffer + length, BUFFER_LEN - length, ":%02x%02x%02x%n", kr, kg, kb, &newlen);
164
165
             length += newlen:
166
167
        return buffer;
168 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



8.29.1.9 int savergb\_kb ( usbdevice \* kb, lighting \* light, int mode )

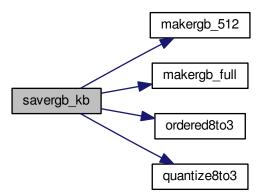
Definition at line 139 of file led keyboard.c.

References usbdevice::dither, usbdevice::fwversion, IS\_STRAFE, makergb\_512(), makergb\_full(), MSG\_SIZE, ordered8to3(), quantize8to3(), and usbsend.

Referenced by cmd\_hwsave\_kb().

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

Here is the call graph for this function:



Here is the caller graph for this function:



8.29.1.10 int savergb\_mouse ( usbdevice \* kb, lighting \* light, int mode )

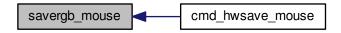
Definition at line 62 of file led\_mouse.c.

References lighting::b, lighting::g, IS\_SABRE, IS\_SCIMITAR, LED\_DPI, LED\_MOUSE, MSG\_SIZE, lighting::r, and usbsend.

Referenced by cmd\_hwsave\_mouse().

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

Here is the caller graph for this function:



8.29.1.11 int updatergb\_kb ( usbdevice \* kb, int force )

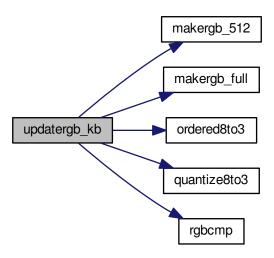
Definition at line 77 of file led\_keyboard.c.

References usbdevice::active, usbprofile::currentmode, usbdevice::dither, lighting::forceupdate, IS\_FULLRANGE, usbprofile::lastlight, usbmode::light, makergb\_512(), makergb\_full(), MSG\_SIZE, ordered8to3(), usbdevice::profile, quantize8to3(), rgbcmp(), lighting::sidelight, and usbsend.

77

```
78
       if(!kb->active)
79
           return 0;
       lighting* lastlight = &kb->profile->lastlight;
80
       lighting* newlight = &kb->profile->currentmode->
81
      light;
       // Don't do anything if the lighting hasn't changed
82
       if(!force && !lastlight->forceupdate && !newlight->forceupdate
83
84
                && !rgbcmp(lastlight, newlight) && lastlight->sidelight == newlight->
       sidelight) // strafe sidelights
8.5
           return 0;
       lastlight->forceupdate = newlight->forceupdate = 0;
86
87
       if(IS_FULLRANGE(kb)){
88
89
            // Update strafe sidelights if necessary
90
            if(lastlight->sidelight != newlight->sidelight) {
                uchar data_pkt[2][MSG_SIZE] = {
91
                      { 0x07, 0x05, 0x08, 0x00, 0x00 },
{ 0x07, 0x05, 0x02, 0, 0x03 }
92
93
94
                 if (newlight->sidelight)
96
                      data_pkt[0][4]=1;
                                             // turn on
97
                 if(!usbsend(kb, data_pkt[0], 2))
98
                     return -1;
99
100
             // 16.8M color lighting works fine on strafe and is the only way it actually works
             uchar data_pkt[12][MSG_SIZE] = {
101
102
                  // Red
                 { 0x7f, 0x01, 0x3c, 0 },
{ 0x7f, 0x02, 0x3c, 0 },
103
104
105
                 \{ 0x7f, 0x03, 0x18, 0 \},
                 { 0x07, 0x28, 0x01, 0x03, 0x01, 0},
106
107
                 // Green
108
                 { 0x7f, 0x01, 0x3c, 0 },
109
                 { 0x7f, 0x02, 0x3c, 0 },
                 { 0x7f, 0x03, 0x18, 0 },
{ 0x07, 0x28, 0x02, 0x03, 0x01, 0},
110
111
112
                  // Blue
                 { 0x7f, 0x01, 0x3c, 0 },
113
114
                 { 0x7f, 0x02, 0x3c, 0 },
115
                  { 0x7f, 0x03, 0x18, 0 },
                  { 0x07, 0x28, 0x03, 0x03, 0x02, 0}
116
117
             makergb_full(newlight, data_pkt);
118
             if(!usbsend(kb, data_pkt[0], 12))
119
120
                 return -1;
121
        } else {
122
            // On older keyboards it looks flickery and causes lighting glitches, so we don't use it.
123
             uchar \ data\_pkt[5][MSG\_SIZE] \ = \ \{
                 { 0x7f, 0x01, 60, 0 },
{ 0x7f, 0x02, 60, 0 },
124
125
                  { 0x7f, 0x03, 60, 0 },
126
                  { 0x7f, 0x04, 36, 0 },
{ 0x07, 0x27, 0x00, 0x00, 0xD8 }
127
128
129
            makergb_512(newlight, data_pkt, kb->dither ?
130
      ordered8to3 : quantize8to3);
131
            if(!usbsend(kb, data_pkt[0], 5))
132
                 return -1;
133
134
        memcpy(lastlight, newlight, sizeof(lighting));
135
136
        return 0;
137 }
```

Here is the call graph for this function:



#### 8.29.1.12 int updatergb\_mouse ( usbdevice \* kb, int force )

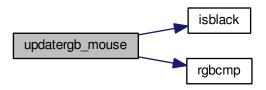
Definition at line 20 of file led\_mouse.c.

References usbdevice::active, lighting::b, usbprofile::currentmode, lighting::forceupdate, lighting::g, isblack(), usbprofile::lastlight, LED\_MOUSE, usbmode::light, MSG\_SIZE, N\_MOUSE\_ZONES, usbdevice::profile, lighting::r, rgbcmp(), and usbsend.

```
2.0
        if(!kb->active)
21
22
            return 0;
        lighting* lastlight = &kb->profile->lastlight;
23
24
        lighting* newlight = &kb->profile->currentmode->
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;
       lastlight->forceupdate = newlight->forceupdate = 0;
31
        \ensuremath{//} Send the RGB values for each zone to the mouse
       uchar data_pkt[2][MSG_SIZE] = {
    { 0x07, 0x22, N_MOUSE_ZONES, 0x01, 0 }, // RGB colors
    { 0x07, 0x05, 0x02, 0 } // Lighting or
32
33
                                                           // Lighting on/off
34
35
36
        uchar* rgb_data = &data_pkt[0][4];
        for(int i = 0; i < N_MOUSE_ZONES; i++) {
   *rgb_data++ = i + 1;
   *rgb_data++ = newlight->r[LED_MOUSE + i];
37
38
39
             *rgb_data++ = newlight->g[LED_MOUSE + i];
40
             *rgb_data++ = newlight->b[LED_MOUSE + i];
41
43
       // Send RGB data
44
       if(!usbsend(kb, data_pkt[0], 1))
45
            return -1:
       int was_black = isblack(kb, lastlight), is_black = isblack(kb, newlight);
46
        if(is_black){
           // If the lighting is black, send the deactivation packet (M65 only)
49
             if(!usbsend(kb, data_pkt[1], 1))
       return -1;
} else if(was_black || force){
50
51
             // If the lighting WAS black, or if we're on forced update, send the activation packet
52
53
            data_pkt[1][4] = 1;
             if(!usbsend(kb, data_pkt[1], 1))
```

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

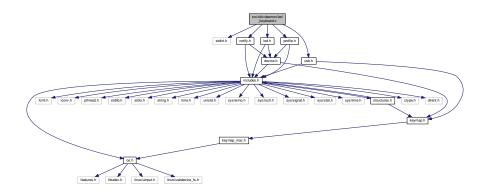
Here is the call graph for this function:



# 8.30 src/ckb-daemon/led\_keyboard.c File Reference

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

Include dependency graph for led keyboard.c:



#### **Macros**

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

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

### **Functions**

- static uchar ordered8to3 (int index, uchar value)
- static uchar quantize8to3 (int index, uchar value)
- static void makergb\_512 (const lighting \*light, uchar data\_pkt[5][64], uchar(\*ditherfn)(int, uchar))
- static void makergb\_full (const lighting \*light, uchar data\_pkt[12][64])
- static int rgbcmp (const lighting \*lhs, const lighting \*rhs)
- int updatergb\_kb (usbdevice \*kb, int force)
- int savergb\_kb (usbdevice \*kb, lighting \*light, int mode)
- int loadrgb\_kb (usbdevice \*kb, lighting \*light, int mode)

### **Variables**

 $((((((\ 0\ )\ \&\ 0xaa) >> \ 1)\ |\ (((\ 0\ )\ \&\ 0x55) << \ 1))\ \&\ 0x33) << \ 2))\ \&\ 0xf0) >> \ 4)\ |\ (((((((((\ 0\ )\ \&\ 0xaa) >> \ 1)\ A))\ |\ A))\ |\ A))$  $\big| \; (((0\ 0\ )\ \&\ 0x55) << 1))\ \&\ 0xcc) >> 2)\ \big| \; ((((((0\ )\ \&\ 0xaa) >> 1)\ \big|\ ((((0\ )\ \&\ 0x55) << 1))\ \&\ 0x33) << 2))\ \&\ 0x33) << 2))\ \&\ 0x55) << 1)$ 0x0f <<4)), (((((((((((0)+1)&0xaa)>>1)|((((0)+1)&0x55)<<1))&0xcc)>>2)|((((((0)+1)&0x55)<<1))&0xcc)>>2)|(((((((0)+1)&0x55)<<1))&0xcc)>>2)|((((((0)+1)&0x55)<<1))&0xcc)>>2)|((((((0)+1)&0x55)<<1))&0xcc)>>2)|((((((0)+1)&0x55)<0x55)<<1))&0xcc)>>2)|((((((0)+1)&0x55)<0x55)<0x55)<0x55)<0x55) 1) & 0xaa) >> 1 | ((( ( 0 ) + 1 ) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((( ( 0 ) + 1 ) & 0xaa) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((( ( 0 ) + 2 ) & 0xaa) >> 1) | ((( ( 0 ) + 2 ) & 0x55) << 1)) & 0x45) << 1) | ((( ( 0 ) + 2 ) & 0x55) << 1)) & 0x45) << 1) | (( ( 0 ) + 2 ) & 0x55) << 1)) & 0x45) << 1) | (( ( 0 ) + 2 ) & 0x55) << 1)) & 0x45) << 1) | (( ( 0 ) + 2 ) & 0x55) << 1) | ( ( ( 0 ) + 2 ) & 0x55) << 1)) & 0x45) << 1) | ( ( ( 0 ) + 2 ) & 0x55) << 1) | ( ( ( 0 ) + 2 ) & 0x55) << 1) | ( ( 0 ) + 2 ) & 0x55) << 1) | ( ( ( 0 ) + 2 ) & 0x55) << 1) | ( ( 0 ) + 2 ) & 0x55) << 1) | ( ( 0 ) + 2 ) & 0x55) << 1) | ( ( 0 ) + 2 ) & 0x55) << 1) | ( ( 0 ) + 2 ) & 0x55) << 1) | ( ( 0 ) + 2 ) & 0x55) << 1) | ( ( 0 ) + 2 ) & 0x55) << 1) | ( ( 0 ) + 2 ) & 0x55) << 1) | ( ( 0 ) + 2 ) & 0x55) << 1) | ( ( 0 ) + 2 ) & 0x55) << 1) | ( ( 0 ) + 2 ) & 0x55) << 1) | ( ( 0 ) + 2 ) & 0x55) << 1) | ( ( 0 ) + 2 ) & 0x55) << 1) | ( ( 0 ) + 2 ) & 0x55) << 1) | ( ( 0 ) + 2 ) & 0x55) << 1) | ( ( 0 ) + 2 ) & 0x55) << 1) | ( ( 0 ) + 2 ) & 0x55) << 1) | ( ( 0 ) + 2 ) & 0x55) << 1) | ( ( 0 ) + 2 ) & 0x55) << 1) | ( ( 0 ) + 2 ) & 0x55) << 1) | ( ( 0 ) + 2 ) & 0x55) << 1) | ( ( 0 ) + 2 ) & 0x55) << 1) | ( ( 0 ) + 2 ) & 0x55) << 1) | ( ( 0 ) + 2 ) & 0x55) << 1) | ( ( 0 ) + 2 ) & 0x55) << 1) | ( ( 0 ) + 2 ) & 0x55) << 1) | ( ( 0 ) + 2 ) & 0x55) << 1) | ( ( 0 ) + 2 ) & 0x55) << 1) | ( ( 0 ) + 2 ) & 0x55) << 1) | ( ( 0 ) + 2 ) & 0x55) << 1) | ( ( 0 ) + 2 ) & 0x55) << 1) | ( ( 0 ) + 2 ) & 0x55) << 1) | ( ( 0 ) + 2 ) & 0x55) << 1) | ( ( 0 ) + 2 ) & 0x55) << 1) | ( 0 ) + 2 ) & 0x55) << 1) | ( 0 ) + 2 ) & 0x55) << 1) | ( 0 ) + 2 ) & 0x55) << 1) | ( 0 ) + 2 ) & 0x55) << 1) | ( 0 ) + 2 ) & 0x55) << 1) | ( 0 ) + 2 ) | ( 0 ) + 2 | ( 0 ) + 2 | ( 0 ) + 2 | ( 0 ) + 2 | ( 0 ) + 2 | ( 0 ) + 2 | ( 0 ) + 2 | ( 0 ) + 2 | ( 0 ) + 2 | ( 0 ) + 2 | ( 0 ) + 2 | ( 0 ) + 2 | ( 0 ) + 2 | ( 0 ) + 2 | ( 0 ) + 2 | ( 0 ) + 2 | ( 0 ) + 2 | ( 0 ) + 2 | ( 0 ) + 2 | ( 0 ) + 2 | ( 0 ) + 2 | ( 0 0xcc) >> 2 | (((((( ( 0 ) + 2 ) & 0xaa) >> 1 ) | ((( ( 0 ) + 2 ) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4 | 1) |((((0) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((0) + 2) + 2) + 1) & 0xaa) >> 1) | $(((\ (\ (\ 0\ )\ +\ 2\ )\ +\ 1\ )\ \&\ 0x55)<<1))\ \&\ 0xcc)>>2)\ \big|\ (((((\ (\ (\ 0\ )\ +\ 2\ )\ +\ 1\ )\ \&\ 0xaa)>>1)\ \big|\ (((\ (\ (\ 0\ )\ +\ 2\ )\ +\ 1\ )\ \&\ 0xaa)>>1)\ \big|\ (((\ (\ (\ (\ 0\ )\ +\ 2\ )\ +\ 1\ )\ \&\ 0xaa)>>1)\ \big|\ (((\ (\ (\ 0\ )\ +\ 2\ )\ +\ 1\ )\ \&\ 0xaa)>>1)\ \big|\ (((\ (\ (\ 0\ )\ +\ 2\ )\ +\ 1\ )\ \&\ 0xaa)>>1)\ \big|\ (((\ (\ (\ 0\ )\ +\ 2\ )\ +\ 1\ )\ \&\ 0xaa)>>1)\ \big|\ (((\ (\ (\ 0\ )\ +\ 2\ )\ +\ 1\ )\ \&\ 0xaa)>>1)\ \big|\ (((\ (\ (\ 0\ )\ +\ 2\ )\ +\ 1\ )\ \&\ 0xaa)>>1)\ \big|\ (((\ (\ (\ 0\ )\ +\ 2\ )\ +\ 1\ )\ \&\ 0xaa)>>1)\ \big|\ (((\ (\ (\ 0\ )\ +\ 2\ )\ +\ 1\ )\ \&\ 0xaa)>>1)\ \big|\ (((\ (\ (\ 0\ )\ +\ 2\ )\ +\ 1\ )\ \&\ 0xaa)>>1)\ \big|\ (((\ (\ (\ 0\ )\ +\ 2\ )\ +\ 1\ )\ \&\ 0xaa)>>1)\ \big|\ (((\ (\ (\ 0\ )\ +\ 2\ )\ +\ 1\ )\ \&\ 0xaa)>>1)\ \big|\ (((\ (\ (\ 0\ )\ +\ 2\ )\ +\ 1\ )\ \&\ 0xaa)>>1)\ \big|\ (((\ (\ (\ 0\ )\ +\ 2\ )\ +\ 1\ )\ \&\ 0xaa)>>1)\ \big|\ (((\ (\ (\ 0\ )\ +\ 2\ )\ +\ 1\ )\ \&\ 0xaa)>>1)\ \big|\ (((\ (\ (\ 0\ )\ +\ 2\ )\ +\ 1\ )\ \&\ 0xaa)>>1)\ \big|\ (((\ (\ (\ 0\ )\ +\ 2\ )\ +\ 1\ )\ \&\ 0xaa)>>1)\ \big|\ (((\ (\ (\ 0\ )\ +\ 2\ )\ +\ 1\ )\ \&\ 0xaa)>>1)\ \big|\ (((\ (\ 0\ )\ +\ 2\ )\ +\ 1\ )\ \&\ 0xaa)>>1)\ \big|\ (((\ (\ 0\ )\ +\ 2\ )\ +\ 1\ )\ \&\ 0xaa)>>1)\ \big|\ (((\ (\ 0\ )\ +\ 2\ )\ +\ 1\ )\ \&\ 0xaa)>>1)\ \big|\ (((\ (\ 0\ )\ +\ 2\ )\ +\ 1\ )\ \&\ 0xaa)>>1)\ \big|\ (((\ (\ 0\ )\ +\ 2\ )\ +\ 1\ )\ \&\ 0xaa)>>1)\ \big|\ (((\ (\ 0\ )\ +\ 2\ )\ +\ 1\ )\ \&\ 0xaa)>>1)\ \big|\ (((\ (\ 0\ )\ +\ 2\ )\ +\ 1\ )\ \&\ 0xaa)>>1)\ \big|\ (((\ (\ 0\ )\ +\ 2\ )\ +\ 1\ )\ \&\ 0xaa)>>1)\ \big|\ (((\ (\ 0\ )\ +\ 2\ )\ +\ 1\ )\ \&\ 0xaa)>>1)\ \big|\ (((\ (\ 0\ )\ +\ 2\ )\ +\ 1\ )\ \&\ 0xaa)>>1)\ \big|\ (((\ (\ 0\ )\ +\ 1\ )\ )\ ((\ (\ 0\ )\ +\ 1\ )\ )\ )\ (((\ (\ 0\ )\ +\ 1\ )\ )\ )\ ((\ (\ 0\ )\ +\ 1\ )\ )\ )\ ((\ (\ 0\ )\ )\ )\ ((\ (\ 0\ )\ )\ )\ ((\ (\ 0\ )\ )\ )\ ((\ (\ 0\ )\ )\ )\ ((\ (\ 0\ )\ )\ )\ ((\ (\ 0\ )\ )\ )\ ((\ (\ 0\ )\ )\ )\ ((\ (\ 0\ )\ )\ )\ ((\ (\ 0\ )\ )\ )\ ((\ (\ 0\ )\ )\ )\ ((\ (\ 0\ )\ )\ )\ ((\ (\ 0\ )\ )\ )\ ((\ (\ 0\ )\ )\ )\ ((\ (\ 0\ )\ )\ )\ ((\ (\ 0\ )\ )\ )\ ((\ (\ 0\ )\ )\ )\ ((\ (\ 0\ )\ )\ )\ ((\ (\ 0\ )\ )\ )\ ((\ (\ 0\ )$ + 1 ) & 0x55) << 1)) & 0xcc) >> 2) | (((((( ( ( 0 ) + 4 ) + 1 ) & 0xaa) >> 1) | ((( ( ( 0 ) + 4 ) + 1 ) & 0x55) <<1)) & 0xcc) >> 2) | (((((((0)+4)+1) & 0xaa) >> 1) | (((((0)+4)+1) & 0x55) << 1)) & 0x33) << 1 $(\ (\ 0\ )+4\ )+2\ )+1\ )\ \&\ 0xaa)>>1)\ |\ (((\ (\ (\ 0\ )+4\ )+2\ )+1\ )\ \&\ 0x55)<<1))\ \&\ 0xcc)>>2)\ |\ (((((\ (\ (\ 0\ )+4\ )+2\ )+1\ )\ \&\ 0x55)<<1))$ )+4)+2)+1) & 0xaa) >> 1) | (((((((0)+4)+2)+1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)),+ 8 ) & 0x55) << 1)) & 0xcc) >> 2) | (((((( ( 0 ) + 8 ) & 0xaa) >> 1) | ((( ( 0 ) + 8 ) & 0x55) << 1)) & 0x33) >> 2) | (((((( ( ( 0 ) + 8 ) + 1 ) & 0xaa) >> 1) | ((( ( ( 0 ) + 8 ) + 1 ) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) (0) + (0) $(0) + 8 + 2 \times 0$   $(0) + 8 \times 0$ ((0)+8)+2) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((((((0)+8)+2)+2)+1) & 0xaa) >>

```
0xaa) >> 1) \mid (((\ (\ (\ 0\ )\ +\ 8\ )\ +\ 2\ )\ +\ 1\ )\ \&\ 0x55) << 1))\ \&\ 0x33) << 2))\ \&\ 0x0f) << 4)), (((((((((((((\ (\ 0\ )\ +\ 8\ )\ +\ 2\ )\ +\ 1\ )\ )\ \&\ 0x55) << 1))))))
(0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (0) + (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) |
1) \mid (((((((0) + 16) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((((0) + 16) + 16) + 2) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 
(((((0) + 16) + 4) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((0) + 16) + 4) & 0xaa) >> 4) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16
0xaa) >> 1 | ((( ( ( ( 0 ) + 16 ) + 4 ) + 1 ) & 0x55) << 1)) & 0xcc) >> 2) | ((((( ( ( ( 0 ) + 16 ) + 4 ) + 1 ) & 0x5)
0xaa) >> 1 | ((( ( ( ( 0 ) + 16 ) + 4 ) + 1 ) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((( ( ( ( 0 ) + 16 ) + 4 ) + 1 ) + 1 ) + 1 ) + 1 ) + 1 ) + 1 ) + 1 ) + 1 )
16\ )+4\ )+2\ )\ \&\ 0xaa)>>1)\ \big|\ (((\ (\ (\ 0\ )+16\ )+4\ )+2\ )\ \&\ 0x55)<<1))\ \&\ 0xcc)>>2)\ \big|\ (((((\ (\ (\ 0\ )+16\ )+4\ )+2\ ))\ \&\ 0x55)<<1))\ .
16\ )+4\ )+2\ )\ \&\ 0xaa)>>1)\ |\ (((\ (\ (\ 0\ )+16\ )+4\ )+2\ )\ \&\ 0x55)<<1))\ \&\ 0x33)<<2))\ \&\ 0xf0)>>4)
2) | (((((((((((((0) + 16) + 4) + 2) & 0xaa) >> 1) | (((((((0) + 16) + 4) + 2) & 0x55) << 1)) & 0x33) << 1)
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)\ )+\ 8)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\ )+\ 16)\
```

```
|16| + 8| + 1| & 0xaa| >> 1 | ((( ( ( ( 0 ) + 16 ) + 8 ) + 1 ) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4)
0xcc) >> 2 | (((((((((0) + 16) + 8) + 2) & 0xaa) >> 1) | ((((((0) + 16) + 8) + 2) & 0x55) << 1)) &
(((0) + 16) + 8) + 2) + 1) & 0x55 < (1)) & 0xcc > 2) | ((((((((0) + 16) + 8) + 2) + 1) & 0xaa) > (((0) + 16) + 8) + 2) + 1) & 0xaa)
((((0) + 16) + 8) + 2) + 1) & 0xaa) >> 1) | (((((((0) + 16) + 8) + 2) + 1) & 0x55) << 1)) & 0x33)
<< 1)) & 0xcc) >> 2) | (((((( ( ( ( 0 ) + 16 ) + 8 ) + 4 ) & 0xaa) >> 1) | ((( ( ( ( 0 ) + 16 ) + 8 ) + 4 ) & 0x55)
 \$ \ 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) <<
(((0) + 16) + 8) + 4) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((((((((0) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 1
) + 16 ) + 8 ) + 4 ) + 2 ) \& 0xaa) >> 1) \mid ((( ( ( ( ( ( 0 ) + 16 ) + 8 ) + 4 ) + 2 ) \& 0x55) << 1)) \& 0x33) << 2))
16\ ) + 8\ ) + 4\ ) + 2\ ) + 1\ ) \ \& \ Oxaa) >> 1)\ \big|\ (((\ (\ (\ (\ 0\ ) + 16\ ) + 8\ ) + 4\ ) + 2\ ) + 1\ ) \ \& \ Ox55) << 1))\ \& \ Oxcc)
& 0x55) <<1)) & 0x33) <<2)) & 0x0f) <<4), ((((((((((0) + 32) & 0xaa) >> 1) | (((((0) + 32) & 0x55) + (0) + 32) & 0x55) |
0xf0)>>4) \mid ((((((((((0)+32)&0xaa)>>1)))(((((0)+32)&0x55)<<1))&0xcc)>>2)))((((((0)+32)&0x55)<<1))&0xcc)>>2)))
) & 0xaa) >> 1) | ((( ( ( 0 ) + 32 ) + 1 ) & 0x55) << 1)) & 0xcc) >> 2) | ((((( ( ( 0 ) + 32 ) + 1 ) & 0xaa) >>
1) & 0x55 (< 1)) & 0xcc (((((((((0)+32)+2)+1) & 0xaa) >> 1) | (((((((0)+32)+2)+1)
(0) + 32 + 4 \times (0) \times (
```

) + 32 ) + 4 ) + 2 ) + 1 ) & 0xaa) >> 1) | ((( ( ( ( ( 0 ) + 32 ) + 4 ) + 2 ) + 1 ) & 0x55) << 1)) & 0xcc) >> 2)1) |(((((((0)+32)+8)+1) & 0x55) << 1)) & 0xcc) >> 2) |(((((((((0)+32)+8)+1) & 0xaa) >> 1) $\big| \; (((\ (\ (\ (\ 0\ )\ +\ 32\ )\ +\ 8\ )\ +\ 1\ )\ \&\ 0x55) <<\ 1))\ \&\ 0x33) <<\ 2))\ \&\ 0x0f) <<\ 4)), \; (((((((((\ (\ (\ 0\ )\ +\ 32\ )\ +\ 8\ )\ +\ 2\ )\ +\ 2\ )\ +\ 3)\ +\ 3)\ +\ 3)$ ) & 0xaa) >> 1) | ((( ( ( ( 0 ) + 32 ) + 8 ) + 2 ) & 0x55) << 1)) & 0xcc) >> 2) | ((((( ( ( ( 0 ) + 32 ) + 8 ) + 2 )  $\$ \ 0 \times aa) >> 1) \ \big| \ \big( ((\ (\ (\ 0\ ) + 32\ ) + 8\ ) + 2\ ) \ \$ \ 0 \times 55) << 1)) \ \$ \ 0 \times 33) << 2)) \ \$ \ 0 \times f0) >> 4) \ \big| \ \big( (((((((\ (\ (\ 0\ ) + 32\$ 32) + 8) + 2) & 0xaa) >> 1) | ((((((0) + 32) + 8) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)),0xcc) >> 2 | (((((((((0)+32)+8)+2)+1)&0xaa) >> 1) | ((((((0)+32)+8)+2)+1)&0x55) 4) & 0xaa >> 1) | ((((((0) + 32) + 8) + 4) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((0) + 32) + 8) + 4) + 4) + 4) + 4) + 4) 32) + 8) + 4) & 0xaa >> 1 | ((( ( ( ( 0 ) + 32 ) + 8 ) + 4 ) & 0x55 >< 1)) & 0x33 >< 2)) & 0x0f >< 4)), 32) + 8) + 4) + 1) & 0x55 << 1)) & 0xcc) >> 2) | ((((((((((((0) + 32) + 8) + 4) + 1) & 0xaa) >> 1) | (((( )+32)+8)+4)+2) & 0xaa) >> 1) | (((((((0)+32)+8)+4)+2) & 0x55) << 1)) & 0x33) << 2))(0.0000) > (0.0000) > (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000)16) & 0xaa) >> 1) | (((((0) + 32) + 16) & <math>0x55) << 1)) & <math>0xcc) >> 2) | (((((((0) + 32) + 16) & <math>0xaa)  $\big| \; (((\ (\ (\ 0\ )\ +\ 32\ )\ +\ 16\ )\ \&\ 0x55) <<\ 1))\ \&\ 0x33) <<\ 2))\ \&\ 0x0f) <<\ 4)), \; (((((((((\ (\ (\ 0\ )\ +\ 32\ )\ +\ 16\ )\ +\ 1\ )\ \&\ 1)\ )\ \&\ 1)$ 0xaa) >> 1 | ((( ( ( ( 0 ) + 32 ) + 16 ) + 1 ) & 0x55) << 1)) & 0xcc) >> 2 | ((((( ( ( ( 0 ) + 32 ) + 16 ) + 1 ) 0) + 32) + 16) + 1) & 0xaa) >> 1) | ((((((0) + 32) + 16) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) $0xaa) >> 1) \mid (((\ (\ (\ (\ 0\ )\ +\ 32\ )\ +\ 16\ )\ +\ 2\ )\ +\ 1\ )\ \&\ 0x55) << 1))\ \&\ 0x33) << 2))\ \&\ 0xf0) >> 4) \mid (((((((((((\ (\ (\ 0\ )\ )\ +\ 32\ )\ +\ 16\ )\ +\ 2\ )\ +\ 1\ )\ \&\ 0x55) << 1))$ ((0) + 32) + 16) + 2) + 1) & 0xaa) >> 1) | ((((((0) + 32) + 16) + 2) + 1) & 0x55) << 1)) & 0xcc)

```
|(((((((0)+32)+16)+4) \& 0x55) << 1)) \& 0xcc) >> 2) |(((((((0)+32)+16)+4) \& 0xaa) >> 1)) |
+32) +16) +4) +1) & 0xaa) >>1) | (((((((0) +32) +16) +4) +1) & 0x55) <<1)) & 0x33) <<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) >>
(0.055) << 1) (0.055) << 1) (0.055) << 1) (0.055) << 1) (0.055) << 1) (0.055) << 1) (0.055) << 1) (0.055) << 1) (0.055) << 1) (0.055) << 1) (0.055) << 1) (0.055) << 1) (0.055) << 1) (0.055) << 1) (0.055) << 1) (0.055) << 1)
1) |(((((((((0)+32)+16)+4)+2)+1) & 0x55) << 1)) & 0xcc) >> 2) |((((((((((0)+32)+16)+16)+16)+16)+16)+16)+16)+16) |
+\ 16\ )\ +\ 8\ )\ \&\ 0xaa)>>1)\ \big|\ (((\ (\ (\ 0\ )\ +\ 32\ )\ +\ 16\ )\ +\ 8\ )\ \&\ 0x55)<<1))\ \&\ 0xcc)>>2)\ \big|\ ((((((\ (\ (\ 0\ )\ )\ +\ 32\ )\ +\ 16\ )\ +\ 8\ )\ \&\ 0x55)<<1))
) + 16 ) + 8 ) & 0xaa) >> 1) | ((( ( ( ( 0 ) + 32 ) + 16 ) + 8 ) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) |
2) \mid ((((((((0)+32)+16)+8) \& 0xaa) >> 1) \mid ((((((0)+32)+16)+8) \& 0x55) << 1)) \& 0x33) << 1)
0xaa) >> 1 | ((((((0)+32)+16)+8)+1)&0x55) << 1)) & 0xcc) >> 2) | ((((((((0)+32)+16)+32)+16)+32)+16)+32)+16) |
+ 8 ) + 1 ) & 0xaa) >> 1) | ((( ( ( ( ( ( 0 ) + 32 ) + 16 ) + 8 ) + 1 ) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) <<
((((0)+32)+16)+8)+2)+1) & 0xaa)>>1) |(((((((0)+32)+16)+8)+2)+1) & 0x55)<<
32) + 16) + 8) + 4) & 0xaa) >> 1) | ((( ((((0) + 32) + 16) + 8) + 4) & 0x55) << 1)) & 0x33) << 2)) & 0x33
0xcc) >> 2 | (((((((((0)+32)+16)+8)+4)+1)&0xaa)>> 1) | (((((((0)+32)+16)+8)+8)+8)+8)+8) |
) + 16) + 8) + 4) + 2) & 0xaa) >> 1) | (((((((0) + 32) + 16) + 8) + 4) + 2) & 0x55) << 1)) & 0x33)
+\ 16\ )\ +\ 4\ )\ +\ 2\ )\ \&\ 0x55) <<\ 1))\ \&\ 0xcc)>>\ 2)\ \big|\ ((((((\ (\ (\ (\ 0\ )\ +\ 32\ )\ +\ 16\ )\ +\ 8\ )\ +\ 4\ )\ +\ 2\ )\ \&\ 0xaa)
```

```
0) + 32) + 16) + 8) + 4) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((0 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(1 + 1)(
0xaa) >> 1 | ((( ( 0 ) + 64 ) & 0x55) << 1)) & 0xcc) >> 2 | ((((( ( 0 ) + 64 ) & 0xaa) >> 1) | ((( ( 0 ) + 64 )
(0.055) << 1) (0.055) << 1) (0.055) << 1) (0.055) << 1) (0.055) << 1)
<<1)) & 0xcc) >>2) | (((((( ( 0 ) + 64 ) & 0xaa) >>1) | ((( ( 0 ) + 64 ) & 0x55) <<1)) & 0x33) <<2)) &
|((((((((0)+64)+1)&0xaa)>>1)|(((((0)+64)+1)&0x55)<<1))&0x33)<<2))&0xf0)>>4)|
(0) + 64 + 2 \times 0 \times 0 \times 0 > 1 ((((((0) + 64) + 2) \times 0 \times 5) < (1)) \times 0 \times 0 > 2) (((((((((0) + 64) + 2) \times 0 \times 5) < (1)) \times 0 \times 0 > 2))
2 ) & 0xaa) >> 1) | ((( ( ( 0 ) + 64 ) + 2 ) & 0x55) << 1)) & 0xcc) >> 2) | ((((( ( ( 0 ) + 64 ) + 2 ) & 0xaa) >>
0xaa) >> 1 | ((( ( ( ( 0 ) + 64 ) + 2 ) + 1 ) & 0x55) << 1)) & 0xcc) >> 2) | ((((( ( ( ( 0 ) + 64 ) + 2 ) + 1 ) & 0x5) << 1)) & 0xcc)
4) + 1) & 0xaa >> 1 | ((( ( ( ( 0 ) + 64 ) + 4 ) + 1 ) & 0x55 >< 1)) & 0xcc >> 2) | ((((( ( ( 0 ) + 64 ) + 4 ) + 4 ) + 1 ) & 0x55 >< 1)) & 0xcc >> 2)
)+64)+4)+1) & 0xaa) >> 1) | ((((((0)+64)+4)+1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 1)
1)) & 0xcc) >> 2 | ((((((((0) + 64) + 4) + 2) & 0xaa) >> 1) | ((((((0) + 64) + 4) + 2) & 0x55) <<
+64)+4)+2)+1) & 0xaa) >> 1) | ((((((0)+64)+4)+2)+1) & 0x55) << 1)) & 0x33) << 2)) & 0x33) << 1) | ((((0)+64)+4)+2)+1) & 0x55) << 1) | (((0)+64)+4)+2)+1) | (((0)+64)+4)+2)+1) | (((0)+64)+4)+2)+1) | (((0)+64)+4)+2)+1) | (((0)+64)+4)+2)+1) | (((0)+64)+4)+2)+1) | (((0)+64)+4)+2)+1) | (((0)+64)+4)+2)+1) | (((0)+64)+4)+2)+1) | (((0)+64)+4)+2)+1) | (((0)+64)+4)+2)+1) | (((0)+64)+4)+2)+1) | (((0)+64)+4)+2)+1) | (((0)+64)+4)+2)+1) | (((0)+64)+4)+2)+1) | (((0)+64)+4)+2)+1) | (((0)+64)+4)+2)+1) | (((0)+64)+4)+2)+1) | (((0)+64)+4)+2)+1) | (((0)+64)+4)+2)+1) | (((0)+64)+4)+2)+1) | (((0)+64)+4)+2)+1) | (((0)+64)+4)+2)+1) | (((0)+64)+4)+2)+1) | (((0)+64)+4)+2)+1) | (((0)+64)+4)+2)+1) | (((0)+64)+4)+2)+1) | ((0)+64)+4)+1) | ((0)+64)+4)+10 | ((0)+64)+4) | ((0)+64)+4) | ((0)+64)+4) | ((0)+64)+4) | ((0)+64)+4) | ((0)+64)+4) | ((0)+64)+4) | ((0)+64)+4) | ((0)+64)+4) | ((0)+64)+4) | ((0)+64)+4) | ((0)+64)+4) | ((0)+64)+4) | ((0)+64)+4) | ((0)+64)+4) | ((0)+64)+4) | ((0)+64)+4) | ((0)+64)+4) | ((0)+64)+4) | ((0)+64)+4) | ((0)+64)+4) | ((0)+64)+4) | ((0)+64)+4) | ((0)+64)+4) | ((0)+64)+4) | ((0)+64)+4) | ((0)+64)+4) | ((0)+64)+4) | ((0)+64)+4) | ((0)+64)+4) | ((0)+64)+4) | ((0)+64)+4) | ((0)+64)+4) | ((0)+64)+4) | ((0)+64)+4) | ((0)+64)+4) | ((0)+64)+4) | ((0)+64)+4) | ((0)+64)+4) | ((0)+64)+4) | ((0)+64)+4) | ((0)+64)+4) | ((0)+64)+4) | ((0)+64)+4) | ((0)+64)+4) | ((0)+64)+4) | ((0)+64)+4) | ((0)+64)+4) | ((0)+64)+4) | ((0)+64)+4) | ((0)+64)+4) | ((0)+64)+4) | ((0)+64)+4) | ((0)+64)+4) | ((0)+64)+4) | ((0)+64)+4) | ((0)+64)+4) | ((0)+64)+4) | ((0)+64)+4) | ((0)+64)+4) | ((0)+64)+4) | ((0)+64)+4) | ((0)+64)+4) | ((0)+64)+4) | ((0)+64)+4) | ((0)+64)+4) | ((0)+64)+4) | ((0)+64)+4) | ((0)+64)+4) | ((0)+64)+4) | ((0)+64)+4) | ((0)+64)+4) | ((0)+64)+4) | ((0)+64)+4) | ((0)+64)+4) | ((0)+64)+4) | ((0)+64)+4) | ((0)+64)+4) | ((0)+64)+4) | ((0)+64)+4) | ((0)+64)+4) | ((0)+64)+4) | ((0)+64)+4) | ((0)+64)+4) | ((0)+64)+4) | ((0)+64)+4) | ((0)+64)+4) | ((0)+64)+4) | ((0)+64)+4) | ((0)+64)+4) | ((0)+64)+4) | ((0)+64)+4) | ((0)+64)+4) | ((0)+64
(\ 0\ )+64\ )+8\ )+1\ )\ \&\ 0xaa)>>1)\ \big|\ (((\ (\ (\ 0\ )+64\ )+8\ )+1\ )\ \&\ 0x55)<<1))\ \&\ 0xcc)>>2)\ \big|\ (((((\ (\ (\ 0\ )+64\ )+8\ )+1\ )\ \&\ 0x55)<<1))\ \&\ 0xcc)>>2)\ \big|\ (((((\ (\ (\ 0\ )+64\ )+8\ )+1\ )\ \&\ 0x55)<<1))\ \&\ 0xcc)>>2)\ \big|\ (((((\ (\ 0\ )+64\ )+8\ )+1\ )\ \&\ 0x55)<<1))\ \&\ 0xcc)>>2)\ \big|\ (((((\ (\ 0\ )+64\ )+8\ )+1\ )\ \&\ 0x55)<<1))\ \&\ 0xcc)>>2)\ \big|\ (((((\ (\ 0\ )+64\ )+8\ )+1\ )\ \&\ 0x55)<<1))\ \&\ 0xcc)>>2)\ \big|\ (((((\ (\ 0\ )+64\ )+8\ )+1\ )\ \&\ 0x55)<<1))\ \&\ 0xcc)>>2)\ \big|\ (((((\ (\ 0\ )+64\ )+8\ )+1\ )\ \&\ 0x55)<<1))\ \&\ 0xcc)>>2)\ \big|\ ((((\ (\ 0\ )+64\ )+8\ )+1\ )\ \&\ 0x55)<<1)
) + 64 + 8 + 1 + 3 \times 0xaa >> 1 + (((((((0) + 64) + 8) + 1) \times 0x55) << 1)) \times 0x33) << 2)) \times 0xf0) >> 1 + (((((((0) + 64) + 8) + 1) \times 0x55) << 1)) \times 0x33) << 2)
(((0) + 64) + 8) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((0) + 64) + 8) + 2) + 1) & 0xaa)) | (((0) + 64) + 8) + 2) + 1) & 0xaa)
((((0)+64)+8)+2)+1) & 0xaa) >> 1) | (((((0)+64)+8)+2)+1) & 0x55) << 1)) & 0x33)
<<1)) & 0xcc) >>2) | ((((((((0)+64)+8)+4)&0xaa)>>1) | (((((0)+64)+8)+4)&0x55)
```

```
(\ 0\ )+64\ )+8\ )+4\ )+1\ )\ \&\ 0xaa)>>1)\ \big|\ \big(((\ (\ (\ (\ 0\ )+64\ )+8\ )+4\ )+1\ )\ \&\ 0x55)<<1))\ \&\ 0xcc)>>2)
+4)+2) & Oxaa) >> 1) | ((( ( ( ( ( 0 ) +64 ) +8 ) +4 ) +2 ) & Ox55) << 1)) & Ox33) << 2)) & Ox0f) <<
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))&0x55)<=1)) & 0x55) = (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) + (0.5) 
0xco) >> 2) \mid ((((((((0)+64)+16)&0xaa)>> 1) \mid (((((0)+64)+16)&0x55)<< 1)) & 0x33) << 1)
<<1)) & 0xcc) >>2) | ((((((((0)+64)+16)+1) & 0xaa) >>1) | ((((((0)+64)+16)+1) & 0x55)
(((((((0)+64)+16)+2) \& 0x55) << 1)) \& 0xcc) >> 2) | ((((((((0)+64)+16)+2) \& 0xaa) >> 1) |
(0) + 64 + 16 + 2 + 1 & 0xaa > 
) \; \& \; 0xaa) >> 1) \; | \; (((\; (\; (\; (\; 0\; )\; +\; 64\; )\; +\; 16\; )\; +\; 4\; )\; \& \; 0x55) << 1)) \; \& \; 0xcc) >> 2) \; | \; (((((\; (\; (\; (\; 0\; )\; +\; 64\; )\; +\; 16\; )\; +\; 4\; )\; )\; ) \; | \; (((\; (\; (\; (\; 0\; )\; +\; 64\; )\; +\; 16\; )\; +\; 4\; )\; ) \; | \; ((\; (\; (\; (\; 0\; )\; +\; 64\; )\; +\; 16\; )\; +\; 4\; )\; ) \; | \; ((\; (\; (\; (\; 0\; )\; +\; 64\; )\; +\; 16\; )\; +\; 4\; )\; ) \; | \; ((\; (\; (\; 0\; )\; +\; 64\; )\; +\; 16\; )\; +\; 4\; )\; | \; ((\; (\; 0\; )\; +\; 64\; )\; +\; 16\; )\; ) \; | \; ((\; (\; (\; 0\; )\; +\; 64\; )\; +\; 16\; )\; )\; | \; ((\; 0\; )\; +\; 64\; )\; | \; ((\; 0\; )\; +\; 64\; )\; | \; ((\; 0\; )\; +\; 64\; )\; )\; | \; ((\; 0\; )\; +\; 64\; )\; | \; ((\; 0\; )\; +\; 64\; )\; | \; (\; 0\; )\; | \; (\; 0\; )\; | \; (\; 0\; )\; | \; (\; 0\; )\; | \; (\; 0\; )\; | \; (\; 0\; )\; | \; (\; 0\; )\; | \; (\; 0\; )\; | \; (\; 0\; )\; | \; (\; 0\; )\; | \; (\; 0\; )\; | \; (\; 0\; )\; | \; (\; 0\; )\; | \; (\; 0\; )\; | \; (\; 0\; )\; | \; (\; 0\; )\; | \; (\; 0\; )\; | \; (\; 0\; )\; | \; (\; 0\; )\; | \; (\; 0\; )\; | \; (\; 0\; )\; | \; (\; 0\; )\; | \; (\; 0\; )\; | \; (\; 0\; )\; | \; (\; 0\; )\; | \; (\; 0\; )\; | \; (\; 0\; )\; | \; (\; 0\; )\; | \; (\; 0\; )\; | \; (\; 0\; )\; | \; (\; 0\; )\; | \; (\; 0\; )\; | \; (\; 0\; )\; | \; (\; 0\; )\; | \; (\; 0\; )\; | \; (\; 0\; )\; | \; (\; 0\; )\; | \; (\; 0\; )\; | \; (\; 0\; )\; | \; (\; 0\; )\; | \; (\; 0\; )\; | \; (\; 0\; )\; | \; (\; 0\; )\; | \; (\; 0\; )\; | \; (\; 0\; )\; | \; (\; 0\; )\; | \; (\; 0\; )\; | \; (\; 0\; )\; | \; (\; 0\; )\; | \; (\; 0\; )\; | \; (\; 0\; )\; | \; (\; 0\; )\; | \; (\; 0\; )\; | \; (\; 0\; )\; | \; (\; 0\; )\; | \; (\; 0\; )\; | \; (\; 0\; )\; | \; (\; 0\; )\; | \; (\; 0\; )\; | \; (\; 0\; )\; | \; (\; 0\; )\; | \; (\; 0\; )\; | \; (\; 0\; )\; | \; (\; 0\; )\; | \; (\; 0\; )\; | \; (\; 0\; )\; | \; (\; 0\; )\; | \; (\; 0\; )\; | \; (\; 0\; )\; | \; (\; 0\; )\; | \; (\; 0\; )\; | \; (\; 0\; )\; | \; (\; 0\; )\; | \; (\; 0\; )\; | \; (\; 0\; )\; | \; (\; 0\; )\; | \; (\; 0\; )\; | \; (\; 0\; )\; | \; (\; 0\; )\; | \; (\; 0\; )\; | \; (\; 0\; )\; | \; (\; 0\; )\; | \; (\; 0\; )\; | \; (\; 0\; )\; | \; (\; 0\; )\; | \; (\; 0\; )\; | \; (\; 0\; )\; | \; (\; 0\; )\; | \; (\; 0\; )\; | \; (\; 0\; )\; | \; (\; 0\; )\; | \; (\; 0\; )\; | \; (\; 0\; )\; | \; (\; 0\; )\; | \; (\; 0\; )\; | \; (\; 0\; )\; | \; (\; 0\; )\; | \; (\; 0\; )\; | \; (\; 0\; )\; | \; (\; 0\; )\; | \; (\; 0\; )\; | \; (\; 0\; )\; | \; (\; 0\; )\; | \; (\; 0\; )\; | \; (\; 0\; )\; | \; (\; 0\; )\; | \; (\; 0\; )\; | \; (\; 0\; )\; | \; (\; 0\; )\; | \; (\; 0\; )\; | \; (\; 0\; )\; | 
0) + 64) + 16) + 4) & 0xaa) >> 1) | ((((((0) + 64) + 16) + 4) & 0x55) << 1)) & 0x33) << 2)) & 0x0f)
<<1)) & 0xcc) >>2) | (((((((((0)+64)+16)+4)+1) & 0xaa) >>1) | (((((((0)+64)+16)+4)+16)+4)
(((0) + 64) + 16) + 4) + 2) & 0xaa >> 1) | ((((((0) + 64) + 16) + 4) + 2) & 0x55) << 1)) & 0xcc)
+64) +16) +4) +2) & 0x55) <<1)) & 0xcc) >>2) | (((((((((0)+64)+16)+4)+2) & 0xaa) >>1)
\big| \; (((\ (\ (\ (\ 0\ )\ +\ 64\ )\ +\ 16\ )\ +\ 4\ )\ +\ 2\ )\ \&\ 0x55) <<1))\ \&\ 0x33) <<2))\ \&\ 0x0f) <<4)), \; ((((((((((\ (\ (\ (\ 0\ )\ )\ +\ 64\ )\ +\ 16\ )\ +\ 4\ )\ +\ 2\ )\ \&\ 0x55) <<1))
) + 16 \ ) + 4 \ ) + 2 \ ) + 1 \ ) \ \& \ 0 xaa) >> 1) \ | \ (((\ (\ (\ (\ (\ 0\ )\ + 64\ ) + 16\ ) + 4\ ) + 2\ ) + 1\ ) \ \& \ 0 x55) << 1)) \ \& \ 0 xcc)
1) |(((((((((0)+64)+16)+4)+2)+1) & 0x55) << 1)) & 0xcc) >> 2) |((((((((((0)+64)+16)+16)+16)+16)+16)+16)+16)+16) |
(0.0033) << 2) (0.0033) << 2) (0.0033) << 2) (0.0033) <> 2) (0.0033) << 2) (0.0033) << 2) (0.0033) << 2) (0.0033) << 2)
```

```
(\ (\ 0\ )\ +\ 64\ )\ +\ 16\ )\ +\ 8\ )\ +\ 1\ )\ \&\ 0xaa)>>1)\ \big|\ (((\ (\ (\ (\ 0\ )\ +\ 64\ )\ +\ 16\ )\ +\ 8\ )\ +\ 1\ )\ \&\ 0x55)<<1))\ \&\ 0xcc)>>
((0) + 64) + 16) + 8) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((((((((0) + 64) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16)
64) + 16) + 8) + 2) & 0xaa) >> 1) | ((( ((((0) + 64) + 16) + 8) + 2) & 0x55) << 1)) & 0x33) << 2)) &
+\ 16\ ) + 8\ ) + 2\ ) + 1\ )\ \&\ Oxaa) >> 1)\ \big|\ (((\ (\ (\ (\ (\ 0\ ) + 64\ ) + 16\ ) + 8\ ) + 2\ ) + 1\ )\ \&\ Ox55) << 1))\ \&\ Oxcc) >> 1)
& 0xaa >> 1 | ((( ( ( ( ( ( ( 0 ) + 64 ) + 16 ) + 8 ) + 4 ) + 1 ) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4 |
+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) << 0
0\ ) + 64\ ) + 16\ ) + 8\ ) + 4\ ) + 2\ ) \ \& \ 0x55) <<1)) \ \& \ 0xcc) >>2)\ \big|\ ((((((\ (\ (\ (\ 0\ )\ + 64\ ) + 16\ ) + 8\ ) + 4\ ) + 2\ )
((((0)+64)+16)+8)+4)+2)+1) & 0xaa) >> 1) |(((((((0)+64)+16)+8)+4)+2)+1) & 0xaa) >> 1) |((((((0)+64)+16)+8)+4)+2)+1) & 0xaa) >> 1) |((((((0)+64)+16)+8)+8)+4)+2)+1) & 0xaa) >> 1) |(((((((((0)+64)+16)+8)+4)+2)+1)+8)+4)+2)+1)
32) & 0xaa) >> 1) | (((((0) + 64) + 32) & <math>0x55) << 1)) & <math>0xcc) >> 2) | (((((((0) + 64) + 32) & <math>0xaa) & 0xaa)
0xaa) >> 1) \mid (((((((0) + 64) + 32) \& 0x55) << 1)) \& 0xcc) >> 2) \mid ((((((((((0) + 64) + 32) \& 0xaa) >> 1) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) 
0xaa) >> 1 | ((( ( ( ( 0 ) + 64 ) + 32 ) + 1 ) & 0x55) << 1)) & 0xcc) >> 2 | ((((( ( ( ( 0 ) + 64 ) + 32 ) + 1 )
0) + 64) + 32) + 1) & 0xaa) >> 1) | ((((((0) + 64) + 32) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f)
0xcc) >> 2 | ((((((((0)+64)+32)+2)&0xaa)>> 1) | (((((0)+64)+32)+2)&0x55)<<1)) &
((0) + 64) + 32) + 2) + 1) & 0xaa) >> 1) | ((((((0) + 64) + 32) + 2) + 1) & 0x55) << 1)) & 0xcc)
|(((((((0)+64)+32)+4) \& 0x55) << 1)) \& 0xcc) >> 2) |(((((((0)+64)+32)+4) \& 0xaa) >> 1)) |
+64)+32)+4)+1) & 0xaa) >> 1) | (((((((0)+64)+32)+4)+1) & 0x55) << 1)) & 0x33) << 2))
```

(0.016) > (0.0+4)+2) & 0xaa >> 1) | ((( ( ( ( ( 0 ) + 64 ) + 32 ) + 4 ) + 2 ) & 0x55 << 1)) & 0x33 << 2)) & 0xf0 >> (0.055) << 1) (0.055) << 1) (0.055) << 1) (0.055) << 1) (0.055) << 1) (0.055) << 1) (0.055) << 1) (0.055) << 1) (0.055) << 1) (0.055) << 1) (0.055) << 1) (0.055) << 1) (0.055) << 1) (0.055) << 1) (0.055) << 1) (0.055) << 1) $4\ ) + 2\ ) + 1\ ) \ \& \ 0xaa) >> 1)\ \big|\ (((\ (\ (\ (\ 0\ ) + 64\ ) + 32\ ) + 4\ ) + 2\ ) + 1\ ) \ \& \ 0x55) << 1))\ \& \ 0x33) << 2))\ \& \ 0x33) << 2))\ \& \ 0x33) << 2)$ ((((0) + 64) + 32) + 4) + 2) + 1) & 0x55 << 1) & 0x33 << 2) & 0x0f << 4), (((((((((((((((0) + 64)) + 32 ) + 8 ) & 0xaa) >> 1) | ((( ( ( ( 0 ) + 64 ) + 32 ) + 8 ) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | 2) | ((((((((0)+64)+32)+8) & 0xaa) >> 1) | ((((((0)+64)+32)+8) & 0x55) << 1)) & 0x33) << 1) + 8 ) + 1 ) & 0xaa) >> 1) | ((( ( ( ( ( 0 ) + 64 ) + 32 ) + 8 ) + 1 ) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) <<((((0)+64)+32)+8)+2)+1) & 0xaa) >> 1) | (((((((0)+64)+32)+8)+2)+1) & 0x55) <<8) + 2) + 1) & 0x55 < (1) & 0x33 < (2) & 0xf0 > (1) & 0x55 < (2) & 0xf0 > (1) & 0x55 < (2) & 0xf0 > (3) & 0xf0 > (4) & 0x55 < (4) & 0x55 < (5) & 0xf0 > (4) & 0) & 0xaa) >> 1 | ((( ( ( ( ( ( 0 ) + 64 ) + 32 ) + 8 ) + 2 ) + 1 ) & 0x55) << 1)) & 0xcc) >> 2 | ((((( ( ( ( ( ( 0 ) + 64 ) + 32 ) + 8 ) + 2 ) + 1 ) & 0x55) << 1)) ) + 32 ) + 8 ) + 4 ) + 1 ) & 0xaa) >> 1) | ((( ( ( ( ( 0 ) + 64 ) + 32 ) + 8 ) + 4 ) + 1 ) & 0x55) << 1)) & 0xcc)) + 64) + 32) + 8) + 4) + 2) + 1) & 0xaa) >> 1) | ((((((0) + 64) + 32) + 8) + 4) + 2) + 1) & 0x55)0xaa) >> 1 | ((( ( ( ( 0 ) + 64 ) + 32 ) + 16 ) & 0x55 | << 1)) & 0xcc |>> 2) | ((((( ( ( ( 0 ) + 64 ) + 32 ) + 16 )  $\$ \ 0 \times aa) >> 1) \ | \ (((\ (\ (\ 0\ )\ +\ 64\ )\ +\ 32\ )\ +\ 16\ )\ \$ \ 0 \times 55) << 1))\ \$ \ 0 \times 33) << 2))\ \$ \ 0 \times fo) >> 4) \ | \ ((((((((\ (\ (\ 0\ )\ +\ 64\ )\ +\ 32\ )\ +\ 16\ )\ \$ \ 0 \times 55) << 1))\ \$ \ 0 \times 33) << 2))\ \$ \ 0 \times fo) >> 4) \ | \ (((((((\ (\ 0\ )\ +\ 64\ )\ +\ 32\ )\ +\ 16\ )\ \$ \ 0 \times 55) << 1))\ \$ \ 0 \times 33) << 2))\ \$ \ 0 \times fo) >> 4) \ | \ (((((((\ (\ 0\ )\ +\ 64\ )\ +\ 32\ )\ +\ 16\ )\ \$ \ 0 \times 55) << 1))\ \$ \ 0 \times 33) << 2))\ \$ \ 0 \times 50) >> 4) \ | \ (((((((\ (\ 0\ )\ +\ 64\ )\ +\ 32\ )\ +\ 16\ )\ 8\ 0 \times 55) << 1))\ \$ \ 0 \times 33) << 2))\ \$ \ 0 \times 50) >> 4) \ | \ (((((((\ (\ 0\ )\ +\ 64\ )\ +\ 32\ )\ +\ 16\ )\ 8\ 0 \times 55) << 1))\ \$ \ 0 \times 33) << 2)$ ) + 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)) & 0x33) <= 0

(0.055) << 1) (0.055) << 1) (0.055) << 1) (0.055) << 1) (0.055) << 1) (0.055) << 1) (0.055) << 1) (0.055) << 1) $\$ \ 0xaa) >> 1) \ | \ (((\ (\ (\ (\ 0\ )\ +\ 64\ )\ +\ 32\ )\ +\ 16\ )\ +\ 1\ )\ \$ \ 0x55) << 1)) \ \$ \ 0xcc) >> 2) \ | \ (((((\ (\ (\ (\ 0\ )\ +\ 64\ )\ +\ 32\ )\ +\ 16\ )\ +\ 1)\ )\ \$ \ 0x55) << 1)) \ \$ \ 0xcc) >> 2) \ | \ ((((\ (\ (\ (\ 0\ )\ +\ 64\ )\ +\ 32\ )\ +\ 16\ )\ +\ 1)\ )\ \$ \ 0x55) << 1)) \ \$ \ 0xcc) >> 2) \ | \ ((((\ (\ (\ (\ 0\ )\ +\ 64\ )\ +\ 32\ )\ +\ 16\ )\ +\ 1)\ )\ \$ \ 0x55) << 1)$ +32) + 16) + 1) & 0xaa) >> 1) | (((((((0) + 64) + 32) + 16) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xaa) >> 1 | ((( ( ( ( 0 ) + 64 ) + 32 ) + 16 ) + 2 ) & 0x55 | << 1)) & 0xcc | >> 2) | ((((( ( ( ( 0 ) + 64 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) | ) + 16) + 2) & 0xaa) >> 1) | (((((((0) + 64) + 32) + 16) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0x0f)64 ) + 32 ) + 16 ) + 2 ) + 1 ) & 0xaa >> 1) | ((( ( ( ( ( 0 ) + 64 ) + 32 ) + 16 ) + 2 ) + 1 ) & 0x55 << 1)) & >> 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) >>  $0xaa) >> 1) \mid (((\ (\ (\ (\ (\ 0\ )\ +\ 64\ )\ +\ 32\ )\ +\ 16\ )\ +\ 4\ )\ +\ 1\ )\ \&\ 0x55) << 1))\ \&\ 0xcc) >> 2) \mid ((((((\ (\ (\ (\ 0\ )\ +\ 16\ )\ +\ 4\ )\ +\ 1\ )\ )\ \&\ 0x55) << 1))$  $64\ ) + 32\ ) + 16\ ) + 4\ ) + 1\ ) \ \& \ 0xaa) >> 1)\ \big|\ (((\ (\ (\ (\ 0\ ) + 64\ ) + 32\ ) + 16\ ) + 4\ ) + 1\ ) \ \& \ 0x55) << 1))\ \& \ (0x55) << 1))\ \& \ (0x55) << 1)$ +64)+32)+16)+4)+1) & 0x55) << 1)) & 0xcc) >> 2) | (((((((0)+64)+32)+16)+4)+1) $\& \ 0xaa) >> 1) \ \big| \ (((\ (\ (\ (\ 0\ )\ +\ 64\ )\ +\ 32\ )\ +\ 16\ )\ +\ 4\ )\ +\ 1\ )\ \&\ 0x55) << 1)) \ \&\ 0x33) << 2)) \ \&\ 0x0f) << 4)),$ (0.055) << 1) (0.055) << 1) (0.055) << 1) (0.055) << 1) (0.055) << 1) (0.055) << 1) (0.055) << 1) (0.055) << 1)64) + 32) + 16) + 4) + 2) + 1) & 0xaa) >> 1) | (((((((((0) + 64) + 32) + 16) + 4) + 2) + 1) & 0x55)) + 64) + 32) + 16) + 8) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((0) + 64) + 32) + 16) + 8) & 0xaa) $1\ )\ \&\ Oxaa) >> 1)\ \big|\ (((\ (\ (\ (\ (\ 0\ )\ +\ 64\ )\ +\ 32\ )\ +\ 16\ )\ +\ 8\ )\ +\ 1\ )\ \&\ Ox55) << 1))\ \&\ Ox33) << 2))\ \&\ Oxf0) >> 1$ ) + 16) + 8) + 2) & 0xaa) >> 1) | ((((((((0) + 64) + 32) + 16) + 8) + 2) & 0x55) << 1)) & 0xcc) >> 2)(0.055) << (1) (0.0033) << (2) (0.0033) << (2) (0.0033) << (2) (0.0033) << (2) (0.0033) << (2) (0.0033) << (2)+ 16 ) + 8 ) + 2 ) & 0xaa) >> 1) | ((( ( ( ( ( ( ( 0 ) + 64 ) + 32 ) + 16 ) + 8 ) + 2 ) & 0x55) << 1)) & 0x33) << 2)) + 1 ) & 0xaa) >> 1) | ((( ( ( ( ( ( ( ( 0 ) + 64 ) + 32 ) + 16 ) + 8 ) + 2 ) + 1 ) & 0x55) << 1)) & 0x33) << 2)) & 0x55 << 1) 32) + 16) + 8) + 2) + 1) & 0x55 < (1)) & 0xcc) >> 2) | (((((((((0) + 64) + 32) + 16) + 8) + 2) + 16) + 1 ) & 0xaa) >> 1) | ((( ( ( ( ( ( ( ( 0 ) + 64 ) + 32 ) + 16 ) + 8 ) + 2 ) + 1 ) & 0x55) << 1)) & 0x33) << 2)) &  $16\ ) + 8\ ) + 4\ ) \ \&\ 0x55) <<1))\ \&\ 0xcc)>>2)\ \big|\ \big(\big(\big(\big(\big(\ (\ (\ 0\ ) + 64\ ) + 32\ ) + 16\ \big) + 8\ \big) + 4\ \big)\ \&\ 0xaa)>>1\big)$ ) + 64) + 32) + 16) + 8) + 4) & 0xaa) >> 1) | ((((((((0) + 64) + 32) + 16) + 8) + 4) & 0x55) << 1)) & 0x = 0 + 0x = 00xcc) >> 2 | (((((((((0)+64)+32)+16)+8)+4)&0xaa)>> 1) | (((((((0)+64)+32)+16)+32)+16)+32)+16) | +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) & 0xaa) >> 1) | (((((((0)+64)+32)+16)+8)+4)+1) & 0xaa) >> 1) | (((((((0)+64)+32)+16)+8)+16)+8)+16) + 16) +)+8)+4)+2)+1) & 0xaa) >> 1) | ((((((((0)+64)+32)+16)+8)+4)+2)+1) & 0x55) << $127 \ ) \ \& \ Oxaa) >> 1) \ \big| \ (((\ (\ 0\ ) \ +\ 127\ ) \ \& \ Ox55) << 1)) \ \& \ Oxcc) >> 2) \ \big| \ (((((\ (\ 0\ ) \ +\ 127\ ) \ \& \ Oxaa) >> 1) \ \big| \ (((\ (\ 0\ ) \ +\ 127\ ) \ \& \ Oxaa) >> 1) \ \big| \ (((\ (\ 0\ ) \ +\ 127\ ) \ \& \ Oxaa) >> 1) \ \big| \ (((\ (\ 0\ ) \ +\ 127\ ) \ \& \ Oxaa) >> 1) \ \big| \ (((\ (\ 0\ ) \ +\ 127\ ) \ \& \ Oxaa) >> 1) \ \big| \ (((\ (\ 0\ ) \ +\ 127\ ) \ \& \ Oxaa) >> 1) \ \big| \ (((\ (\ 0\ ) \ +\ 127\ ) \ \& \ Oxaa) >> 1) \ \big| \ (((\ (\ 0\ ) \ +\ 127\ ) \ \& \ Oxaa) >> 1) \ \big| \ (((\ (\ 0\ ) \ +\ 127\ ) \ \& \ Oxaa) >> 1) \ \big| \ (((\ (\ 0\ ) \ +\ 127\ ) \ \& \ Oxaa) >> 1) \ \big| \ (((\ (\ 0\ ) \ +\ 127\ ) \ \& \ Oxaa) >> 1) \ \big| \ (((\ (\ 0\ ) \ +\ 127\ ) \ \& \ Oxaa) >> 1) \ \big| \ (((\ (\ 0\ ) \ +\ 127\ ) \ \& \ Oxaa) >> 1) \ \big| \ (((\ (\ 0\ ) \ +\ 127\ ) \ \& \ Oxaa) >> 1) \ \big| \ (((\ (\ 0\ ) \ +\ 127\ ) \ \& \ Oxaa) >> 1) \ \big| \ (((\ (\ 0\ ) \ +\ 127\ ) \ \& \ Oxaa) >> 1) \ \big| \ (((\ (\ 0\ ) \ +\ 127\ ) \ \& \ Oxaa) >> 1) \ \big| \ (((\ (\ 0\ ) \ +\ 127\ ) \ \& \ Oxaa) >> 1) \ \big| \ (((\ (\ 0\ ) \ +\ 127\ ) \ \& \ Oxaa) >> 1) \ \big| \ (((\ (\ 0\ ) \ +\ 127\ ) \ \& \ Oxaa) >> 1) \ \big| \ (((\ (\ 0\ ) \ +\ 127\ ) \ \& \ Oxaa) >> 1) \ \big| \ (((\ (\ 0\ ) \ +\ 127\ ) \ \& \ Oxaa) >> 1) \ \big| \ (((\ (\ 0\ ) \ +\ 127\ ) \ \& \ Oxaa) >> 1) \ \big| \ (((\ (\ 0\ ) \ +\ 127\ ) \ \& \ Oxaa) >> 1) \ \big| \ (((\ (\ 0\ ) \ +\ 127\ ) \ \& \ Oxaa) >> 1) \ \big| \ (((\ (\ 0\ ) \ +\ 127\ ) \ \& \ Oxaa) >> 1) \ \big| \ (((\ (\ 0\ ) \ +\ 127\ ) \ \& \ Oxaa) >> 1) \ \big| \ (((\ (\ 0\ ) \ +\ 127\ ) \ \& \ Oxaa) >> 1) \ \big| \ (((\ (\ 0\ ) \ +\ 127\ ) \ \& \ Oxaa) >> 1) \ \big| \ (((\ (\ 0\ ) \ +\ 127\ ) \ \& \ Oxaa) >> 1) \ \big| \ (((\ (\ 0\ ) \ +\ 127\ ) \ \& \ Oxaa) >> 1) \ \big| \ (((\ (\ 0\ ) \ +\ 127\ ) \ \& \ Oxaa) >> 1) \ \big| \ (((\ (\ 0\ ) \ +\ 127\ ) \ \& \ Oxaa) >> 1) \ \big| \ (((\ (\ 0\ ) \ +\ 127\ ) \ \& \ Oxaa) >> 1) \ \big| \ (((\ (\ 0\ ) \ +\ 127\ ) \ \& \ Oxaa) >> 1) \ \big| \ (((\ (\ 0\ ) \ +\ 127\ ) \ \& \ Oxaa) >> 1) \ \big| \ (((\ (\ 0\ ) \ +\ 127\ ) \ \& \ Oxaa) >> 1) \ \big| \ (((\ (\ 0\ ) \ +\ 127\ ) \ ) \ \ (((\ (\ 0\ ) \ +\ 127\ ) \ ) \ \big| \ (((\ (\ 0\ ) \ +\ 127$ (0) + 127) & 0x55 << 1) & 0x33 << 2) & 0xf0 >> 4 | (((((((((0) + 127) & 0xaa) >> 1) | ((((0) + 127) & 0xaa) >> 1) | ((((0) + 127) & 0xaa) >> 1) | ((((0) + 127) & 0xaa) >> 1) | (((0) + 127) & $127 \cdot ((((((0) + 127) & 0xcc) >> 2) | (((((((0) + 127) & 0xaa) >> 1) | ((((0) + 127) & 0x55) << 1)) & 0x55) << 1)) & 0x55) << 1)$ 1)) & 0xcc) >> 2) | (((((((0) + 127) + 1) & 0xaa) >> 1) | (((((0) + 127) + 1) & 0x55) << 1)) & 0x33) 0xcc) >> 2 | ((((((((0) + 127) + 1) & 0xaa) >> 1) | (((((0) + 127) + 1) & 0x55) << 1)) & 0x33) << 2)) >> 2) | ((((( ( ( 0 ) + 127 ) + 2 ) & 0xaa) >> 1) | ((( ( ( 0 ) + 127 ) + 2 ) & 0x55) << 1)) & 0x33) << 2)) &  $2) \mid ((((((((0) + 127) + 2) & 0xaa) >> 1) \mid (((((0) + 127) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0x0f)$ 0xcc) >> 2 | (((((((((0) + 127) + 2) + 1) & 0xaa) >> 1) | ((((((0) + 127) + 2) + 1) & 0x55) << 1)) & 4) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((0) + 127) + 4) & 0xaa) >> 1) | ((((((0) + 127) + 4) & 0x55) 

 $(\ 0\ ) +\ 127\ ) +\ 8\ ) \ \&\ 0xaa) >>\ 1)\ \big|\ (((\ (\ (\ 0\ ) +\ 127\ ) +\ 8\ ) \ \&\ 0x55) <<\ 1))\ \&\ 0xcc) >>\ 2)\ \big|\ (((((\ (\ (\ 0\ ) +\ 127\ ) +\ 8\ ) +\ 8)) \ \&\ 0x55) <<\ 1))$  $127 + 8 + 1 \times 0$  (((((((0) + 127) + 8) + 1) & 0x55) <<1)) & 0xcc) >>2) | (((((((0) + 127) + 8) + 1) & 0x55) <<1)) & 0xcc) >>2) | (((((((0) + 127) + 8) + 1) & 0x55) <<1))  $127 + 8 + 1 \times 0$  (((((((0) + 127 + 8) + 1) & 0x55) <<1)) & 0x33) <<2)) & 0xf0) >>4) 2) | ((((((((0) + 127) + 8) + 1) & 0xaa) >> 1) | ((((((0) + 127) + 8) + 1) & 0x55) << 1)) & 0x33) << 1) <<1)) & 0xcc) >>2) | ((((((((0) + 127) + 8) + 2) & 0xaa) >>1) | ((((((0) + 127) + 8) + 2) & 0x55) +8)+2) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((0) + 127) + 8) + 2) + 1) & 0xaa) + 127) + 1 ) & 0xaa) >> 1) | ((( ( ( ( ( 0 ) + 127 ) + 8 ) + 2 ) + 1 ) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (0,0) & (0,0) (0+127 + 127+ 127) + 8) + 4) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((0) + 127) + 8) + 4) & 0xaa) >> 1) | ((( ( ( ( 0 ) + 127 ) + 8 ) + 4 ) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((( ( ( ( 0 ) + 127 ) + 127 ) + 127 ) + 127 ) + 127 ) + 127 ) + 127 ) (((0) + 127) + 8) + 4) + 1) & 0xaa) >> 1) | ((((((0) + 127) + 8) + 4) + 1) & 0x55) << 1)) & 0x33)) & 0xaa) >> 1) | (((((((0) + 127) + 8) + 4) + 2) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((0) + 127) + 127) + 127) + 127) + 127) + 127) + 127) +8)+4)+2) & 0xaa) >> 1) | (((((((0)+127)+8)+4)+2) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) $>>1)\mid(((\ (\ (\ (\ (\ 0\ )\ +\ 127\ )\ +\ 8\ )\ +\ 4\ )\ +\ 2\ )\ +\ 1\ )\ \&\ 0x55)<<1))\ \&\ 0xcc)>>2)\mid((((((\ (\ (\ (\ 0\ )\ +\ 127\ )\ +\ 1$ (8) + (4) + (2) + (1) & (3) & (4)+ 127) + 16) + 1) & 0xaa) >> 1) | ((((((0) + 127) + 16) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0)  $0xcc)>>2) \mid ((((((((0)+127)+16)+1) \& 0xaa)>>1) \mid ((((((0)+127)+16)+1) \& 0x55)<<1))$ ((0) + 127) + 16) + 2) + 1) & 0xaa) >> 1) | ((((((0) + 127) + 16) + 2) + 1) & 0x55) << 1)) & 0x33) $(0) + 127 + 16 + 4 \times 0xaa >> 1$   $| ((((((0) + 127) + 16) + 4) \times 0x55) << 1)) \times 0xcc) >> 2)$   $| (((((((0) + 127) + 16) + 4) \times 0x55) << 1)) \times 0xcc) >> 2)$ 

 $(\ (\ (\ 0\ )\ +\ 127\ )\ +\ 16\ )\ +\ 4\ )\ \&\ 0x33) << 2))\ \&\ ((\ (\ (\ 0\ )\ +\ 127\ )\ +\ 16\ )\ +\ 4\ )\ \&\ 0x55) << 1))\ \&\ 0x33) << 2))\ \&\ ((\ (\ 0\ )\ +\ 127\ )\ +\ 16\ )\ +\ 4\ )\ \&\ 0x55) << 1))$ 0xaa) >> 1 | ((( ( ( ( ( 0 ) + 127 ) + 16 ) + 4 ) + 2 ) & 0x55) << 1)) & 0xcc) >> 2 | ((((( ( ( ( ( 0 ) + 127 ) + 127 ) + 127 ) + 127 ) + 127 ) + 127 ) + 127 ) 16) + 4) + 2) & 0xaa) >> 1) | (((((((0) + 127) + 16) + 4) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) $(\ (\ (\ (\ 0\ )\ +\ 127\ )\ +\ 16\ )\ +\ 4\ )\ +\ 2\ )\ +\ 1\ )\ \&\ 0x55)<<1))\ \&\ 0x33)<<2))\ \&\ 0xf0)>>4)\ |\ ((((((((\ (\ (\ 0\ )\ )+\ 127\ )\ +\ 12$  $127\ ) + 16\ ) + 4\ ) + 2\ ) + 1\ ) \ \& \ 0xaa) >> 1)\ \big|\ (((\ (\ (\ (\ 0\ ) + 127\ ) + 16\ ) + 4\ ) + 2\ ) + 1\ ) \ \& \ 0x55) << 1))\ \& \ 0x55) << 1))\ \& \ 0x55) << 1)$ 1) |(((((((0)+127)+16)+8) & 0x55) << 1)) & 0xcc) >> 2) |((((((((0)+127)+16)+8) & 0xaa) + 127) + 16) + 16) $+\ 1\ )\ \&\ 0xaa) >> 1)\ \big|\ (((\ (\ (\ (\ 0\ )\ +\ 127\ )\ +\ 16\ )\ +\ 8\ )\ +\ 1\ )\ \&\ 0x55) <<\ 1))\ \&\ 0x33) <<\ 2))\ \&\ 0x0f) <<\ 4)),$ +2) & 0xaa) >> 1) | ((( ( ( ( ( 0 ) + 127 ) + 16 ) + 8 ) + 2 ) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), + 127 ) + 16 ) + 8 ) + 2 ) + 1 ) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((( ( ( ( ( 0 ) + 127 ) + 16 ) + 16 ) + 127 ) + 16 ) + 127 ) + 16 ) )+8)+2)+1) & 0xaa) >> 1) | (((((((0)+127)+16)+8)+2)+1) & 0x55) << 1)) & 0xcc) >> 2) | $(\ (\ (\ (\ 0\ )\ +\ 127\ )\ +\ 16\ )\ +\ 8\ )\ +\ 4\ )\ \&\ 0x55)<<1))\ \&\ 0xcc)>>2)\ |\ ((((((\ (\ (\ 0\ )\ +\ 127\ )\ +\ 16\ )\ +\ 8\ )\ +\ 4\ )\ \&\ 0x55)<<1)$ (((0) + 127) + 16) + 8) + 4) & 0xaa) >> 1) | ((((((0) + 127) + 16) + 8) + 4) & 0x55) << 1)) & 0xcc)<< 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((((((((0) + 127) + 16) + 8) + 4) + 1) + 1) & 0xaa) >> 1) | (((( + 127) + 16) + 8) + 4) + 1) & 0xaa) >> 1) | (((( + 127) + 16) + 8) + 4) + 1) & 0xaa) >> 1) (((((0) + 127) + 16) + 8) + 4) + 1) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((0) + 127) + 16) + 8) + 16) + 8) + 16) + 127) + 16) + 14) + 1) & 0xaa >> 1 ((( ( ( ( ( ( ( ( ( 0 ) + 127 ) + 16 ) + 8 ) + 4 ) + 1 ) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) ) + 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 > (1) & (0x) < (1) & (0x) < (2) & (+ 16 ) + 8 ) + 4 ) + 2 ) & 0xaa) >> 1) | ((( ( ( ( ( ( 0 ) + 127 ) + 16 ) + 8 ) + 4 ) + 2 ) & 0x55) << 1)) & 0x33) (0) + (127) + (16) ++4)+2)+1) & 0xaa >> 1) | ((( ( ( ( ( ( ( ( 0 ) + 127) + 16 ) + 8 ) + 4 ) + 2 ) + 1 ) & 0x55 >< 1)) & 0x33+4)+2)+1) & 0xaa >> 1) | ((( ( ( ( ( ( ( ( 0 ) + 127 ) + 16 ) + 8 ) + 4 ) + 2 ) + 1 ) & 0x55 > << 1)) & 0x33

<< 2)) & 0xf0) >> 4) | (((((((( ( ( 0 ) + 127 ) + 32 ) & 0xaa) >> 1) | ((( ( ( 0 ) + 127 ) + 32 ) & 0x55) << 1))  $) + 127 + 32 + 1 + 3 \times 0x55 <<1) \times 0xcc >> 2 + 1 \times 0xcc >> 2 \times 0xcc >> 2 \times 0xcc >> 2 \times 0xcc >> 3 \times 0xccc >> 3 \times 0xcc >> 3 \times 0xcc >> 3 \times 0xccc >> 3 \times 0xccc >> 3 \times 0xccc >> 3 \times 0xccc >>$ 0xaa) >> 1 | ((( ( ( ( 0 ) + 127 ) + 32 ) + 2 ) & 0x55) << 1)) & 0xcc) >> 2 | ((((( ( ( ( 0 ) + 127 ) + 32 ) + 2 ) ) + 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)) & 0x33) << 1) | (((0) + 127) + 32) + 2) & 0x55) << 1) | ((0) + 127) + 32) + 2) | ((0) + 127) + 32) + 2) | ((0) + 127) + 32) | ((0) + 127) + 32) | ((0) + 127) + 32) | ((0) + 127) + 32) | ((0) + 127) + 32) | ((0) + 127) + 32) | ((0) + 127) + 32) | ((0) + 127) + 32) | ((0) + 127) + 32) | ((0) + 127) + 32) | ((0) + 127) + 32) | ((0) + 127) + 32) | ((0) + 127) + 32) | ((0) + 127) + 32) | ((0) + 127) + 32) | ((0) + 127) + 32) | ((0) + 127) + 32) | ((0) + 127) + 32) | ((0) + 127) + 32) | ((0) + 127) + 32) | ((0) + 127) + 32) | ((0) + 127) + 32) | ((0) + 127) + 32) | ((0) + 127) + 32) | ((0) + 127) + 32) | ((0) + 127) + 32) | ((0) + 127) + 32) | ((0) + 127) + 32) | ((0) + 127) + 32) | ((0) + 127) + 32) | ((0) + 127) + 32) | ((0) + 127) + 32) | ((0) + 127) + 32) | ((0) + 127) + 32) | ((0) + 127) + 32) | ((0) + 127) + 32) | ((0) + 127) + 32) | ((0) + 127) + 32) | ((0) + 127) + 32) | ((0) + 127) + 32) | ((0) + 127) + 32) | ((0) + 127) + 32) | ((0) + 127) + 32) | ((0) + 127) + 32) | ((0) + 127) + 32) | ((0) + 127) + 32) | ((0) + 127) + 32) | ((0) + 127) + 32) | ((0) + 127) + 32) | ((0) + 127) + 32) | ((0) + 127) + 32) | ((0) + 127) + 32) | ((0) + 127) + 32) | ((0) + 127) + 32) | ((0) + 127) + 32) | ((0) + 127) + 32) | ((0) + 127) + 32) | ((0) + 127) + 32) | ((0) + 127) + 32) | ((0) + 127) + 32) | ((0) + 127) + 32) | ((0) + 127) + 32) | ((0) + 127) + 32) | ((0) + 127) + 32) | ((0) + 127) + 32) | ((0) + 127) + 32) | ((0) + 127) + 32) | ((0) + 127) + 32) | ((0) + 127) + 32) | ((0) + 127) + 32) | ((0) + 127) + 32) | ((0) + 127) + 32) | ((0) + 127) + 32) | ((0) + 127) + 32) | ((0) + 127) + 32) | ((0) + 127) + 32) | ((0) + 127) + 32) | ((0) + 127) + 32) | ((0) + 127) + 32) | ((0) + 127) + 32) | ((0) + 127) + 32) | ((0) + 127) + 32) | ((0) + 127) + 32) | ((0) + 127) + 32) | ((0) + 127) + 32) | ((0) + 127) + 32) | ((0) + 127) + 32) | ((0) + 127) + 32) | ((0) + 127) + 32) | ((0) + 127) + 32) | ((0) + 127) + 32) | ((0) + 127) + 32) | ((00xaa) >> 1 | ((( ( ( ( ( 0 ) + 127 ) + 32 ) + 2 ) + 1 ) & 0x55) << 1)) & 0xcc) >> 2 | ((((( ( ( ( ( 0 ) + 127 ) + 127 ) + 127 ) + 127 ) + 127 ) + 127 ) + 127 ) 0xcc) >> 2 | (((((((((0) + 127) + 32) + 4) & 0xaa) >> 1) | ((((((0) + 127) + 32) + 4) & 0x55) << 1)) (0.033) << 2) (0.033) << 2) (0.033) << 2) (0.033) << 2) (0.033) << 2) (0.033) << 2) (0.033) << 2) (0.033) << 2) (0.033) << 2) (0.033) << 2)) + 1 ) & 0xaa) >> 1) | ((( ( ( ( ( 0 ) + 127 ) + 32 ) + 4 ) + 1 ) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((0) + 127) + 32) + 4) + 2) & 0xaa) >> 1) | ((((((0) + 127) + 32) + 4) + 2) & 0x55) << 1)) & 0x55) << 1)+4)+2)+1) & 0xaa) >> 1) | (((((((0)+127)+32)+4)+2)+1) & 0x55) << 1)) & 0x33) << 2))(0.0000) > (0.0000) > (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000) < (0.0000)+ 127 ) + 32 ) + 8 ) & 0xaa) >> 1) | ((( ( ( ( 0 ) + 127 ) + 32 ) + 8 ) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) 0xco >> 2) | (((((((((0) + 127) + 32) + 8) & 0xaa) >> 1) | ((((((0) + 127) + 32) + 8) & 0x55) << 1)) ((((0) + 127) + 32) + 8) + 1) & 0x55) <<1)) & 0x33) <<2)) & 0xf0) >>4) | (((((((((((((0) + 127 + 127) + ((0) + 127) + 32) + 8) + 1) & 0xaa) >> 1) | ((((((0) + 127) + 32) + 8) + 1) & 0x55) << 1)) & 0x33) $127 + 32 + 3 + 3 + 2 \times 0 \times 0 \times 0 > 1 = ((((((((((0) + 127) + 32) + 32) + 3) + 2) \times 0 \times 5) < (1)) \times 0 \times 3) < (2))$  $)+127\ )+32\ )+8\ )+2\ )+1\ )\ \&\ 0xaa)>>1)\ \big|\ (((\ (\ (\ (\ (\ 0\ )+127\ )+32\ )+8\ )+2\ )+1\ )\ \&\ 0x55)<<1))\ \&$ 

0xaa) >> 1 | ((( ( ( ( ( ( ( 0 ) + 127 ) + 32 ) + 8 ) + 4 ) + 1 ) & 0x55) << 1)) & 0xcc) >> 2 | ((((( ( ( ( ( ( 0 ) + 127 ) + 32 ) 127 + 32 + 3 + 3 + 4 + 1 + 3 & 0xaa >> 1 | ((( ( ( ( ( ( 0 ) + 127 ) + 32 ) + 8 ) + 4 ) + 1 ) & 0x55) << 1)) & +127 + 32 + 3 + 4 + 1 + 1 & 0x55 < 1 & 0x55 < 1 & 0x55 < 1 < 1 < 0x55 < 1 < 1 < 0x55 < 1 < 1 < 0x55 < 1 < 0x $\& 0xaa) >> 1) \mid ((((((((0) + 127) + 32) + 8) + 4) + 1) \& 0x55) << 1)) \& 0x33) << 2)) \& 0x0f) << 4)),$ ((0) + 127) + 32) + 8) + 4) + 2) + 1) & 0xaa) >> 1) | ((((((0) + 127) + 32) + 8) + 4) + 2) + 1) | (((0) + 127) + 32) + $0\ ) + 127\ ) + 32\ ) + 16\ ) \ \& \ 0x55) <<1)) \ \& \ 0xcc) >>2)\ \big|\ (((((\ (\ (\ 0\ )\ +\ 127\ )\ +\ 32\ )\ +\ 16\ )\ \&\ 0xaa) >>1)\ \big|$ ) + 32 ) + 16 ) & 0xaa) >> 1) | ((( ( ( ( 0 ) + 127 ) + 32 ) + 16 ) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) <<16) + 1) & 0xaa) >> 1) | ((( ( ( ( ( 0 ) + 127 ) + 32 ) + 16 ) + 1 ) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) <<<<1)) & 0xcc) >>2) | (((((((((0)+127)+32)+16)+2) & 0xaa) >>1) | (((((((0)+127)+32)+32)+32)+32)+32) + >> 1) | ((( ( ( ( ( ( 0 ) + 127 ) + 32 ) + 16 ) + 2 ) & 0x55) << 1)) & 0xcc) >> 2) | ((((( ( ( ( ( 0 ) + 127 ) + 32 ) + 3 +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) + 12))) + 127) + 12127 + 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) + 0xaa) >> 1 | ((( ( ( ( 0 ) + 127 ) + 32 ) + 16 ) + 4 ) & 0x55 | ( 1) & 0xcc | () ((((( ( ( ( 0 ) + 127 ) + 127 ) + 127 ) + 127 ) + 127 ) +32) +16) +4) & 0xaa) >> 1) | ((( ( ( ( ( 0 ) + 127 ) + 32 ) + 16 ) + 4 ) & 0x55) << 1)) & 0x33) << 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) + 32) + 16) + 32) +)+4 )+1 ) & 0xaa ) >> 1) <math>| ((( ( ( ( ( ( ( 0 ) + 127 ) + 32 ) + 16 ) + 4 ) + 1 ) & 0x55 << 1)) & 0x33 << 2)) &  $(\;(\;0\;)\;+\;127\;)\;+\;32\;)\;+\;16\;)\;+\;4\;)\;+\;2\;)\;\&\;0xaa)>>\;1)\;|\;(((\;(\;(\;0\;)\;+\;127\;)\;+\;32\;)\;+\;16\;)\;+\;4\;)\;+\;2\;)\;\&\;0x55)$ 

```
+ 16 ) + 4 ) + 2 ) + 1 ) & 0xaa) >> 1) | ((( ( ( ( ( ( ( ( 0 ) + 127 ) + 32 ) + 16 ) + 4 ) + 2 ) + 1 ) & 0x55) << 1)) &
) + 127 + 32 + 16 + 8 & 0xaa >> 1  | (((((((0) + 127) + 32) + 16) + 8) & 0x55) << 1)) & 0x33) 
32) + 16) + 8) + 1) & 0xaa) >> 1) | (((((((0) + 127) + 32) + 16) + 8) + 1) & 0x55) << 1)) & 0xcc)
127 + 32 + 16 + 8 + 1 \times 0xaa >> 1 = ((((((((0) + 127) + 32) + 16) + 8) + 1) \times 0x55) << 1)) &
0) + 127) + 32) + 16) + 8) + 4) & 0xaa) >> 1) | ((((((0) + 127) + 32) + 16) + 8) + 4) & 0x55) << 
+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) |
+32) + 16) + 8) + 4) + 1) & 0xaa) >> 1) | (((((((0) + 127) + 32) + 16) + 8) + 4) + 1) & 0x55) <<
+\ 32\ )+\ 16\ )+\ 8\ )+\ 4\ )+\ 2\ )\ \&\ 0xaa)>>1)\ \big|\ (((\ (\ (\ (\ (\ (\ 0\ )+\ 127\ )+\ 32\ )+\ 16\ )+\ 8\ )+\ 4\ )+\ 2\ )\ \&\ 0x55)<<
+32) +16) +8) +4) +2) & 0xaa) >>1) |(((((((((0)+127)+32)+16)+8)+4)+2) & 0x55) <<
 \$ \ 0xaa) >> 1) \ | \ (((\ (\ 0\ )\ +\ 127\ )\ +\ 64\ )\ \$ \ 0x55) << 1))\ \$ \ 0x33) << 2))\ \$ \ 0x0f) << 4)), \ (((((((((\ (\ 0\ )\ +\ 127\ )\ +\ 64\ )\ \$ \ 0x55)\ << 1))\ \$ \ 0x33) << 2))
+ 127 ) + 64 ) + 1 ) & 0xaa) >> 1) | ((( ( ( ( 0 ) + 127 ) + 64 ) + 1 ) & 0x55) << 1)) & 0x33) << 2)) & 0xf0)
```

((0) + 127) + 64) + 2) + 1) & 0xaa >> 1) | (((((((0) + 127) + 64) + 2) + 1) & 0x55) << 1)) & 0x33) $(\ 0\ ) + 127\ ) + 64\ ) + 4\ ) \ \&\ 0xaa) >> 1)\ \big|\ (((\ (\ (\ 0\ ) + 127\ ) + 64\ ) + 4\ ) \ \&\ 0x55) << 1))\ \&\ 0xcc) >> 2)\ \big|\ ((((((\ (\ 0\ ) + 127\ ) + 64\ ) + 4\ ) \ \&\ 0x55) << 1))\ \&\ 0xcc) >> 2)\ \big|\ (((((\ (\ (\ 0\ ) + 127\ ) + 64\ ) + 4\ ) \ \&\ 0x55) << 1))\ \&\ 0xcc) >> 2)\ \big|\ (((((\ (\ (\ 0\ ) + 127\ ) + 64\ ) + 4\ ) \ \&\ 0x55) << 1))\ \&\ 0xcc) >> 2)\ \big|\ (((((\ (\ (\ 0\ ) + 127\ ) + 64\ ) + 4\ ) \ \&\ 0x55) << 1))\ \&\ 0xcc) >> 2)\ \big|\ (((((\ (\ (\ 0\ ) + 127\ ) + 64\ ) + 4\ ) \ \&\ 0x55) << 1))\ \&\ 0xcc) >> 2)\ \big|\ (((((\ (\ (\ 0\ ) + 127\ ) + 64\ ) + 4\ ) \ \&\ 0x55) << 1))\ \&\ 0xcc) >> 2)\ \big|\ (((((\ (\ (\ 0\ ) + 127\ ) + 64\ ) + 4\ ) \ \&\ 0x55) << 1))\ \&\ 0xcc) >> 2)\ \big|\ (((((\ (\ (\ 0\ ) + 127\ ) + 64\ ) + 4\ ) \ \&\ 0x55) << 1))\ \&\ 0xcc) >> 2)\ \big|\ (((((\ (\ (\ 0\ ) + 127\ ) + 64\ ) + 4\ ) \ \&\ 0x55) << 1))\ \&\ 0xcc) >> 2)\ \big|\ (((((\ (\ (\ 0\ ) + 127\ ) + 64\ ) + 4\ ) \ \&\ 0x55) << 1))\ \&\ 0xcc) >> 2)\ \big|\ (((((\ (\ (\ 0\ ) + 127\ ) + 64\ ) + 4\ ) \ \&\ 0x55) << 1))\ \&\ 0xcc) >> 2)\ \big|\ (((((\ (\ (\ 0\ ) + 127\ ) + 64\ ) + 4\ ) \ \&\ 0x55) << 1))\ \&\ 0xcc) >> 2)\ \big|\ (((((\ (\ (\ 0\ ) + 127\ ) + 64\ ) + 4\ ) \ \&\ 0x55) << 1))\ \&\ 0xcc) >> 2)\ \big|\ (((((\ (\ (\ 0\ ) + 127\ ) + 64\ ) + 4\ ) \ \&\ 0x55) << 1))\ \&\ 0xcc) >> 2)\ \big|\ (((((\ (\ (\ 0\ ) + 127\ ) + 64\ ) + 4\ ) \ \&\ 0x55) << 1))\ \&\ 0xcc) >> 2)\ \big|\ (((((\ (\ (\ 0\ ) + 127\ ) + 64\ ) + 4\ ) \ \&\ 0x55) << 1))\ \&\ 0xcc) >> 2)\ \big|\ (((((\ (\ (\ 0\ ) + 127\ ) + 64\ ) + 4\ ) \ \&\ 0x55) << 1))\ \&\ 0xcc) >> 2)\ \big|\ ((((((\ (\ (\ 0\ ) + 127\ ) + 64\ ) + 4\ ) + 4\ ) \ \&\ 0x55) << 1)$  \ ((((((\ (\ (\ 0\ ) + 127\ ) + 64\ ) + 4\ ) + 4\ ) \ \&\ 0x55) << 1) \ (((((\ (\ (\ (\ 0\ ) + 127\ ) + 64\ ) + 4\ ) + 4\ ) \ \&\ 0x55) << 1) \ (((((\ (\ (\ 0\ ) + 127\ ) + 64\ ) + 4\ ) + 4\ ) \ \&\ 0x55) << 1) \ ((((\ (\ (\ 0\ ) + 127\ ) + 64\ ) + 4\ ) + 4\ ) \ \&\ 0x55) << 1) \ (((\ (\ (\ (\ 0\ ) + 127\ ) + 127\ ) + 44\ ) \ \&\ 0x55) << 1) \ (((\ (\ (\ (\ 0\ ) + 127\ ) + 127\ ) + 127\ ) + 44\ ) \ \ ((\ (\ (\ (\ (\ (\ (\ (\ (\ 0\ ) + 127\ ) + 127\ ) + 127\ ) + 44  $(\ (\ (\ 0\ )\ +\ 127\ )\ +\ 64\ )\ +\ 4\ )\ \&\ 0x33)<<2))\ \&\ (((\ (\ (\ 0\ )\ +\ 127\ )\ +\ 64\ )\ +\ 4\ )\ \&\ 0x55)<<1))\ \&\ 0x33)<<2))\ \&\ ((\ (\ (\ 0\ )\ +\ 127\ )\ +\ 64\ )\ +\ 4\ )\ \&\ 0x55)<<1)$ 0xaa) >> 1 | ((( ( ( ( ( 0 ) + 127 ) + 64 ) + 4 ) + 2 ) & 0x55) << 1)) & 0xcc) >> 2 | ((((( ( ( ( ( 0 ) + 127 ) + 127 ) + 127 ) + 127 ) + 127 ) + 127 ) + 127 )  $64\ ) + 4\ ) + 2\ ) \ \& \ 0xaa) >> 1)\ \big|\ (((\ (\ (\ (\ 0\ ) + 127\ ) + 64\ ) + 4\ ) + 2\ ) \ \& \ 0x55) << 1))\ \& \ 0x33) << 2))\ \& \ 0x0f)$ ((((0) + 127) + 64) + 4) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((((((((((0) + 12)) + 12) + $127\ ) + 64\ ) + 4\ ) + 2\ ) + 1\ ) \ \& \ 0xaa) >> 1)\ \big|\ (((\ (\ (\ (\ 0\ ) + 127\ ) + 64\ ) + 4\ ) + 2\ ) + 1\ ) \ \& \ 0x55) << 1))\ \& \ 0x55) << 1))\ \& \ 0x55) << 1)$ 0xcc) >> 2 | ((((((((((0) + 127) + 64) + 4) + 2) + 1) & 0xaa) >> 1) | ((((((((0) + 127) + 64) + 4) + 4) + 4) + 4)  $1)\mid((((\ (\ (\ (\ 0\ )\ +\ 127\ )\ +\ 64\ )\ +\ 8\ )\ \&\ 0x55)<<1))\ \&\ 0xcc)>>2)\mid((((((\ (\ (\ (\ 0\ )\ +\ 127\ )\ +\ 64\ )\ +\ 8\ )\ \&\ 0xaa))>>2)\mid(((((\ (\ (\ (\ 0\ )\ +\ 127\ )\ +\ 64\ )\ +\ 8\ )\ \&\ 0xaa))>>2)\mid((((\ (\ (\ (\ (\ 0\ )\ +\ 127\ )\ +\ 64\ )\ +\ 8\ )\ \&\ 0xaa))>>2)\mid(((\ (\ (\ (\ (\ 0\ )\ +\ 127\ )\ +\ 64\ )\ +\ 8\ )\ \&\ 0xaa))>>2)\mid(((\ (\ (\ (\ (\ 0\ )\ +\ 127\ )\ +\ 64\ )\ +\ 8\ )\ \&\ 0xaa))>>2)\mid((\ (\ (\ (\ (\ 0\ )\ +\ 127\ )\ +\ 64\ )\ +\ 8\ )\ \&\ 0xaa))>>2)\mid((\ (\ (\ (\ (\ 0\ )\ +\ 127\ )\ +\ 64\ )\ +\ 8\ )\ \&\ 0xaa))>>2)\mid((\ (\ (\ (\ (\ 0\ )\ +\ 127\ )\ +\ 64\ )\ +\ 8\ )\ \&\ 0xaa))>>2)\mid((\ (\ (\ (\ (\ 0\ )\ +\ 127\ )\ +\ 64\ )\ +\ 8\ )\ \&\ 0xaa))>>2)\mid((\ (\ (\ (\ (\ 0\ )\ +\ 127\ )\ +\ 64\ )\ +\ 8\ )\ \&\ 0xaa))>>2)\mid((\ (\ (\ (\ (\ 0\ )\ +\ 127\ )\ +\ 64\ )\ +\ 8\ )\ \&\ 0xaa))>>2)\mid((\ (\ (\ (\ (\ 0\ )\ +\ 127\ )\ +\ 64\ )\ +\ 8\ )\ \&\ 0xaa))>>2)\mid((\ (\ (\ (\ (\ 0\ )\ +\ 127\ )\ +\ 64\ )\ +\ 8\ )\ \&\ 0xaa))>>2)\mid((\ (\ (\ (\ (\ 0\ )\ +\ 127\ )\$  $1) \mid ((((((((0) + 127) + 64) + 8) + 1) & 0x55) << 1)) & 0xcc) >> 2) \mid (((((((((0) + 127) + 64) + 8) + 1) + 64) + 8) + 127) + 1$ + 1 ) & 0xaa) >> 1) | ((( ( ( ( ( 0 ) + 127 ) + 64 ) + 8 ) + 1 ) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), + 2 ) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | ((((((( ( ( ( ( 0 ) + 127 ) + 64 ) + 8 ) + 2 ) & 0xaa) >>+2) & 0xaa) >> 1) | ((( ( ( ( ( 0 ) + 127 ) + 64 ) + 8 ) + 2 ) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ) + 8 ) + 2 ) + 1 ) & 0xaa) >> 1) | ((( ( ( ( ( ( ( 0 ) + 127 ) + 64 ) + 8 ) + 2 ) + 1 ) & 0x55) << 1)) & 0xcc) >> 2) |(((0) + 127) + 64) + 8) + 4) & 0xaa) >> 1) | ((((((0) + 127) + 64) + 8) + 4) & 0x55) << 1)) & 0xcc)

```
4\ ) + 1\ ) \ \& \ Oxaa) >> 1)\ \big|\ (((\ (\ (\ (\ (\ 0\ ) + 127\ ) + 64\ ) + 8\ ) + 4\ ) + 1\ ) \ \& \ Ox55) << 1))\ \& \ Ox33) << 2))\ \& \ Oxf0)
) + 64) + 8) + 4) + 2) & 0xaa) >> 1) | (((((((0) + 127) + 64) + 8) + 4) + 2) & 0x55) << 1)) & 0xcc)
64\ ) + 8\ ) + 4\ ) + 2\ ) \ \& \ 0xaa) >> 1)\ \big|\ (((\ (\ (\ (\ (\ 0\ ) + 127\ ) + 64\ ) + 8\ ) + 4\ ) + 2\ ) \ \& \ 0x55) << 1))\ \& \ 0x33) << 1)
127 + 64 + 8 + 4 + 2 + 1 \times 0x55 < 1) & 0xcc >> 2 
2) + 1) & 0xaa >> 1 | ((( ( ( ( ( ( ( ( ( 0 ) + 127 ) + 64 ) + 8 ) + 4 ) + 2 ) + 1 ) & 0x55) << 1)) & 0x33) << 2)) &
& 0xcc >> 2) | ((((((((0) + 127) + 64) + 16) & 0xaa) >> 1) | ((((((0) + 127) + 64) + 16) & 0x55)
127 + 64 + 16 + 2 & 0xaa >> 1  | ((( ( ( ( ( 0 ) + 127 ) + 64 ) + 16 ) + 2 ) & 0x55 ) << 1)) & 0x33 ) << 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 127 + 
+ 127) + 64) + 16) + 2) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((((((((((()) + 127) + 127) + 64) + 127) + 64) + 127) + 64) + 127) + 64) + 64)
16) + 2) + 1) & 0xaa) >> 1) | (((((((((0) + 127) + 64) + 16) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) |
(((0) + 127) + 64) + 16) + 4) & 0xaa) >> 1) | ((((((0) + 127) + 64) + 16) + 4) & 0x55) << 1)) & 0x55) << 1)
1) & 0xaa) >> 1) | ((( ( ( ( ( ( ( 0 ) + 127 ) + 64 ) + 16 ) + 4 ) + 1 ) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >>
127 + 64 + 16 + 4 + 2 \times 0xaa >> 1 | ((((((((0) + 127) + 64) + 16) + 4) + 2) & 0x55) << 1)) &
0) + 127) + 64) + 16) + 4) + 2) & 0xaa) >> 1) | (((((((0) + 127) + 64) + 16) + 4) + 2) & 0x55) <<
+64) +16) +4) +2) +1) & 0xaa) >>1) | (((((((((0)+127)+64)+16)+4)+2)+1) & 0x55) <<
+\ 64\ )\ +\ 16\ )\ +\ 4\ )\ +\ 2\ )\ +\ 1\ )\ \&\ 0xaa) >> 1)\ \big|\ (((\ (\ (\ (\ (\ 0\ )\ +\ 127\ )\ +\ 64\ )\ +\ 16\ )\ +\ 4\ )\ +\ 2\ )\ +\ 1\ )\ \&\ 0x55) <<
```

127) + 64) + 16) + 8) & 0xaa) >> 1) | ((( ( ( ( 0 ) + 127 ) + 64 ) + 16 ) + 8 ) & 0x55) << 1)) & 0xcc) >>+ 16 ) + 8 ) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((( ( ( ( ( 0 ) + 127 ) + 64 ) + 16 ) + 8 ) + 1 ) & 0xaa) >> $(\;(\;0\;)\;+\;127\;)\;+\;64\;)\;+\;16\;)\;+\;8\;)\;+\;2\;)\;\&\;0xaa)>>\;1)\;|\;(((\;(\;(\;0\;)\;+\;127\;)\;+\;64\;)\;+\;16\;)\;+\;8\;)\;+\;2\;)\;\&\;0x55)$ 16) + 8) + 2) & 0xaa) >> 1) | ((((((((0) + 127) + 64) + 16) + 8) + 2) & 0x55) << 1)) & 0xcc) >> 2) |(0.055) << 1) (0.055) << 1) (0.055) << 1) (0.055) << 1) (0.055) << 1) (0.055) << 1) (0.055) << 1) (0.055) << 1)((0) + 127) + 64) + 16) + 8) + 2) + 1) & 0xaa) >> 1) | ((((((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) + 16127 + 64 + 16 + 8 + 4 & 0xaa >> 1 | ((( ( ( ( ( ( 0 ) + 127 ) + 64 ) + 16 ) + 8 ) + 4 ) & 0x55 >< 1) \ & 0x55 >< 1 (0.0033) << 2) & 0.0000 >> 4) & 0xaa) >> 1) | ((( ( ( ( ( ( 0 ) + 127 ) + 64 ) + 16 ) + 8 ) + 4 ) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 16) + 8) + 4) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((0) + 127) + 64) + 16) + 8) + 4) + 1)) + 16) + 8) + 4) + 2) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((0) + 127) + 64) + 16) + 8) + 4) + 2)+ 16 ) + 8 ) + 4 ) + 2 ) & 0x55) << 1)) & 0xcc) >> 2) | ((((( ( ( ( ( ( ( 0 ) + 127 ) + 64 ) + 16 ) + 8 ) + 4 ) + 2 ) 8) + 4) + 2) + 1) & 0xaa >> 1 | (((((((((0) + 127) + 64) + 16) + 8) + 4) + 2) + 1) & 0x55) << 1)) (0.033) << 2) (0.033) << 2) (0.033) << 2) (0.033) << 2) (0.033) << 2) (0.033) << 2) (0.033) << 2) (0.033) << 2)0) + 127) + 64) + 16) + 8) + 4) + 2) + 1) & 0xaa) >> 1) | (((((((0) + 127) + 64) + 16) + 8) + 4) + 16) + 1127) + 64) + 32) & 0xaa) >> 1) | ((((((0) + 127) + 64) + 32) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) 127) + 64) + 32) + 1) & 0xaa) >> 1) | (((((((0) + 127) + 64) + 32) + 1) & 0x55) << 1)) & 0x33) <<

```
(((0) + 127) + 64) + 32) + 2) & 0xaa) >> 1) | ((((((0) + 127) + 64) + 32) + 2) & 0x55) << 1)) & 0x55) << 1)
1) & 0xaa >> 1) | ((( ( ( ( ( ( ( 0 ) + 127 ) + 64 ) + 32 ) + 2 ) + 1 ) & 0x55 << 1)) & 0x33 << 2)) & 0xf0 >>
+ 1 ) & 0x55) << 1)) & 0xcc) >> 2) | ((((( ( ( ( ( ( 0 ) + 127 ) + 64 ) + 32 ) + 2 ) + 1 ) & 0xaa) >> 1) | ((( ( ( (
) + 64) + 32) + 4) & 0xaa) >> 1) | (((((((0) + 127) + 64) + 32) + 4) & 0x55) << 1)) & 0xcc) >> 2) |
127) + 64) + 32) + 4) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((0) + 127) + 64) + 32) + 4) & 0xaa) >>
+ 127) + 64) + 32) + 4) + 1) & 0xaa) >> 1) | (((((((0) + 127) + 64) + 32) + 4) + 1) & 0x55) << 1))
0) + 127) + 64) + 32) + 4) + 1) & 0xaa) >> 1) | ((((((0) + 127) + 64) + 32) + 4) + 1) & 0x55) << 
((0) + 127) + 64 + 32 + 4 + 20 & 0x55 < 1) & 0xcc > 2 | (((((((((0) + 127) + 64) + 32) + 64) + 32) + 64) + 32) |
+4)+2) & 0xaa) >> 1) | ((( ( ( ( ( ( ( 0 ) + 127 ) + 64 ) + 32 ) + 4 ) + 2 ) & 0x55) << 1)) & 0x33) << 2)) &
(0) + 127 + 64 + 32 + 4 + 2 + 1 & 0xaa >> 1 & (((((((0) + 127) + 64) + 32) + 4) + 2) + 1
(((((((0) + 127) + 64) + 32) + 4) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((0) + 127) + 127) + 127) + 127) + 127) + 127))
(((0) + 127) + 64) + 32) + 8) & 0xaa) >> 1) | ((((((0) + 127) + 64) + 32) + 8) & 0x55) << 1)) & 0x55) << 1)
0xcc) >> 2 | (((((((((0) + 127) + 64) + 32) + 8) & 0xaa) >> 1) | ((((((0) + 127) + 64) + 32) + 8)
(8.0x55) << 1) (0.0x55) << 1)
((((((((0) + 127) + 64) + 32) + 8) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((0) + 127) + 64) + 32) + 64) + 32) + 64) + 32) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 
8) & 0xaa) >> 1) | (((((((0) + 127) + 64) + 32) + 8) & <math>0x55) << 1)) & <math>0x33) << 2)) & <math>0x0f) << 4)),
+ 127) + 64) + 32) + 8) + 2) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((0) + 127) + 64) + 32) + 8) + 2)
```

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

```
>> 1) | ((( ( ( ( ( ( ( ( ( ( 0 ) + 127 ) + 64 ) + 32 ) + 16 ) + 4 ) + 2 ) + 1 ) & 0x55) << 1)) & 0x33) << 2)) & 0x0f)
\big| \; (((\ (\ (\ (\ (\ 0\ )\ +\ 127\ )\ +\ 64\ )\ +\ 32\ )\ +\ 16\ )\ +\ 8\ )\ \&\ 0x55) <<1))\ \&\ 0x33) <<2))\ \&\ 0xf0)>>4)\ \big| \; (((((((((((\ (\ (\ (\ 0\ )\ )\ +\ 127\ )\ +\ 64\ )\ +\ 32\ )\ +\ 16\ )\ +\ 8\ )\ \&\ 0x55)<<1))
(\ 0\ )+\ 127\ )+\ 64\ )+\ 32\ )+\ 16\ )+\ 8\ )\ \&\ 0xaa)>>1)\ \big|\ (((\ (\ (\ (\ 0\ )+\ 127\ )+\ 64\ )+\ 32\ )+\ 16\ )+\ 8\ )\ \&\ 0x55)
+32)+16)+8)+1) \& 0xaa)>>1) \mid (((((((((0)+127)+64)+32)+16)+8)+1) \& 0x55)<<1))
64) + 32) + 16) + 8) + 1) & 0xaa) >> 1) | (((((((0) + 127) + 64) + 32) + 16) + 8) + 1) & 0x55) <<
0) + 127) + 64) + 32) + 16) + 8) + 2) & 0xaa) >> 1) | ((((((0) + 127) + 64) + 32) + 16) + 8) + 2)
((((0) + 127) + 64) + 32) + 16) + 8) + 2) + 1) & 0xaa) >> 1) | (((((((((0) + 127) + 64) + 32) + 127) + 64) + 32) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127)
+4) & 0xaa) >> 1) | ((( ( ( ( ( ( ( ( 0 ) + 127 ) + 64 ) + 32 ) + 16 ) + 8 ) + 4 ) & 0x55) << 1)) & 0xcc) >> 2) |
)+8)+4) & 0xaa) >> 1) | ((((((((0)+127)+64)+32)+16)+8)+4) & <math>0x55) << 1)) & 0xcc) >> 1)
+16) +8) +4) +1) & 0xaa) >>1) | ((((((((((0) + 127) + 64) + 32) + 16) + 8) + 4) + 1) & 0x55)
(0) + 127 + 64 + 32 + 16 + 8 + 4 + 2 + 1 \times 2000 = 0
16) + 8) + 4) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((((0) + 127) + 64) + 32) + 16) + 8)
) & 0xaa) >> 1) | ((( ( ( ( ( ( ( ( ( ( ( 0 ) + 127 ) + 64 ) + 32 ) + 16 ) + 8 ) + 4 ) + 2 ) + 1 ) & 0x55) << 1)) & 0xcc)
)+127 )+64 )+32 )+16 )+8 )+4 )+2 )+1 ) & 0x55) <<1)) & 0x33) <<2)) & 0x0f) <<4)), }
```

```
8.30.1 Macro Definition Documentation
```

```
8.30.1.1 #define BR1( x) ((((x) & 0x0x3x3) >> 1) | (((x) & 0x5x5) << 1))
```

Definition at line 9 of file led\_keyboard.c.

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

Definition at line 10 of file led keyboard.c.

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

Definition at line 11 of file led\_keyboard.c.

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

Definition at line 12 of file led\_keyboard.c.

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

Definition at line 13 of file led\_keyboard.c.

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

Definition at line 14 of file led\_keyboard.c.

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

Definition at line 15 of file led\_keyboard.c.

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

Definition at line 16 of file led\_keyboard.c.

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

Definition at line 17 of file led\_keyboard.c.

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

Definition at line 18 of file led\_keyboard.c.

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

Definition at line 19 of file led\_keyboard.c.

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

Definition at line 20 of file led\_keyboard.c.

#### 8.30.2 Function Documentation

#### 8.30.2.1 int loadrgb\_kb ( usbdevice \* kb, lighting \* light, int mode )

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

Definition at line 181 of file led\_keyboard.c.

References lighting::b, ckb\_err, usbdevice::fwversion, lighting::g, MSG\_SIZE, N\_KEYS\_HW, lighting::r, usbrecv, and usbsend.

Referenced by hwloadmode().

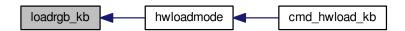
```
182
        if(kb->fwversion >= 0x0120){
            uchar data_pkt[12][MSG_SIZE] = {
183
                 { 0x0e, 0x14, 0x03, 0x01, 0x01, mode + 1, 0x01 }, { 0xff, 0x01, 60, 0 },
184
185
186
                 { 0xff, 0x02, 60, 0 },
                 { 0xff, 0x03, 24, 0
188
                 { 0x0e,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

```
249
              };
250
              uchar in_pkt[4][MSG_SIZE] = {
251
                   { 0xff, 0x01, 60, 0 },
252
                   { 0xff, 0x02, 60, 0 },
253
                   { 0xff, 0x03, 60, 0 },
254
                   { 0xff, 0x04, 36, 0 },
              };
// Write initial packet
256
257
              if(!usbsend(kb, data_pkt[0], 1))
258
                   return -1;
              // Read colors
259
              for (int i = 1; i < 5; i++) {
260
                   if(!usbrecv(kb, data_pkt[i],in_pkt[i - 1]))
261
                        return -1;
263
                   if(memcmp(in_pkt[i - 1], data_pkt[i], 4)){
264
                       ckb_err("Bad input header\n");
265
                        return -1;
266
267
268
              // Copy the data back to the mode
269
              uint8_t mr[N_KEYS_HW / 2], mg[N_KEYS_HW / 2], mb[
       N_KEYS_HW / 2];
270
             memcpy(mr,
                                 in_{pkt[0]} + 4, 60);
              memcpy(mr + 60, in_pkt[1] +
2.71
272
                                in_pkt[1] + 16, 48);
              memcpy(mg,
273
              memcpy(mg + 48, in_pkt[2] + 4, 24);
274
                                in_pkt[2] + 28, 36);
              memcpy(mb,
              memcpy(mb + 36, in_pkt[3] + 4, 36);
275
              // Unpack LED data to 8bpc format
for(int i = 0; i < N_KEYS_HW; i++){
  int    i_2 = i / 2;</pre>
276
277
278
                   uint8_t r, g, b;
280
281
                   \ensuremath{//} 3-bit intensities stored in alternate nybbles.
                  if (i & 1) {
    r = 7 - (mr[i_2] >> 4);
282
283
                       g = 7 - (mg[i\_2] >> 4);

b = 7 - (mb[i\_2] >> 4);
284
285
286
                   } else {
287
                       r = 7 - (mr[i_2] \& 0x0F);
                       g = 7 - (mg[i_2] & 0x0F);

b = 7 - (mb[i_2] & 0x0F);
288
289
290
291
                   // Scale 3-bit values up to 8 bits.
                  light->r[i] = r << 5 | r << 2 | r >> 1;
light->g[i] = g << 5 | g << 2 | g >> 1;
293
294
                   light -> b[i] = b << 5 | b << 2 | b >> 1;
295
296
297
         return 0:
298 }
```

Here is the caller graph for this function:



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

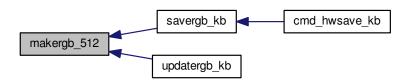
Definition at line 36 of file led keyboard.c.

References lighting::b, lighting::g, N\_KEYS\_HW, and lighting::r.

Referenced by savergb\_kb(), and updatergb\_kb().

```
39
            // Compress RGB values to a 512-color palette
            for (int i = 0; i < N_KEYS_HW; i += 2) {</pre>
                    char r1 = ditherfn(i, light->r[i]), r2 = ditherfn(i + 1, light->r[i + 1]);
41
                   char f1 = ditherfn(i, light->g[i]), g2 = ditherfn(i + 1, light->g[i + 1]);
char g1 = ditherfn(i, light->b[i]), g2 = ditherfn(i + 1, light->g[i + 1]);
char b1 = ditherfn(i, light->b[i]), b2 = ditherfn(i + 1, light->b[i + 1]);
r[i / 2] = (7 - r2) << 4 | (7 - r1);
g[i / 2] = (7 - g2) << 4 | (7 - g1);</pre>
42
4.3
44
45
                    b[i / 2] = (7 - b2) << 4 | (7 - b1);
46
47
            memcpy(data_pkt[0] + 4, r, 60);
memcpy(data_pkt[1] + 4, r + 60, 12);
48
49
            memcpy(data_pkt[1] + 16, g, 48);
memcpy(data_pkt[2] + 4, g + 48, 24);
memcpy(data_pkt[2] + 28, b, 36);
50
51
53
            memcpy(data_pkt[3] + 4, b + 36, 36);
54 }
```

Here is the caller graph for this function:



8.30.2.3 static void makergb\_full ( const lighting \* light, uchar data\_pkt[12][64] ) [static]

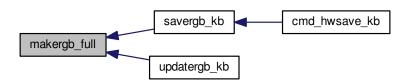
Definition at line 56 of file led\_keyboard.c.

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

Referenced by savergb\_kb(), and updatergb\_kb().

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

Here is the caller graph for this function:



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

Definition at line 24 of file led\_keyboard.c.

References bit\_reverse\_table.

Referenced by savergb\_kb(), and updatergb\_kb().

Here is the caller graph for this function:

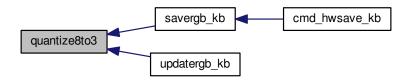


8.30.2.5 static uchar quantize8to3 (int index, uchar value) [static]

Definition at line 32 of file led\_keyboard.c.

Referenced by savergb\_kb(), and updatergb\_kb().

Here is the caller graph for this function:



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

Definition at line 72 of file led\_keyboard.c.

References lighting::b, lighting::g, N\_KEYS\_HW, and lighting::r.

Referenced by updatergb\_kb().

Here is the caller graph for this function:



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

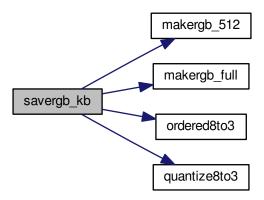
Definition at line 139 of file led keyboard.c.

References usbdevice::dither, usbdevice::fwversion, IS\_STRAFE, makergb\_512(), makergb\_full(), MSG\_SIZE, ordered8to3(), quantize8to3(), and usbsend.

Referenced by cmd\_hwsave\_kb().

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

Here is the call graph for this function:



Here is the caller graph for this function:



### 8.30.2.8 int updatergb\_kb ( usbdevice \*kb, int force )

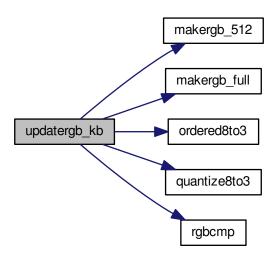
Definition at line 77 of file led\_keyboard.c.

References usbdevice::active, usbprofile::currentmode, usbdevice::dither, lighting::forceupdate, IS\_FULLRANGE, usbprofile::lastlight, usbmode::light, makergb\_512(), makergb\_full(), MSG\_SIZE, ordered8to3(), usbdevice::profile, quantize8to3(), rgbcmp(), lighting::sidelight, and usbsend.

```
78
      if(!kb->active)
79
          return 0;
      lighting* lastlight = &kb->profile->lastlight;
80
      lighting* newlight = &kb->profile->currentmode->
81
     light;
      // Don't do anything if the lighting hasn't changed
82
      if(!force && !lastlight->forceupdate && !newlight->forceupdate
83
84
             && !rgbcmp(lastlight, newlight) && lastlight->sidelight == newlight->
     sidelight)
                // strafe sidelights
85
          return 0:
      lastlight->forceupdate = newlight->forceupdate = 0;
86
87
      if(IS_FULLRANGE(kb)){
88
          // Update strafe sidelights if necessary
          90
91
92
93
94
              if (newlight->sidelight)
```

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

Here is the call graph for this function:



### 8.30.3 Variable Documentation

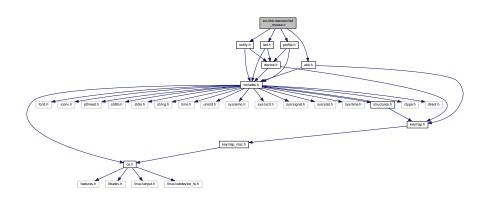
1) | ((( 0 ) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((( 0 ) & 0xaa) >> 1) | ((( 0 ) & 0x55) << 1)) & 0xcc) >> $0\ )+1\ )\ \&\ 0x55)<<1))\ \&\ 0xcc)>>2)\ |\ ((((((\ 0\ )+1\ )\ \&\ 0xaa)>>1)\ |\ ((((\ 0\ )+1\ )\ \&\ 0x55)<<1))\ \&\ 0x33)<<2))\ \&\ 0x33)<<2))\ \&\ 0x33)<<2))\ \&\ 0x33)<<2)$ 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) | 1)) & 0xcc) >> 2) | (((((((0)+2)+1) & 0xaa) >> 1) | (((((0)+2)+1) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 1 $0xaa) >> 1) \mid (((((0)+2)+1) \& 0x55) << 1)) \& 0x33) << 2)) \& 0x0f) << 4)), (((((((((0)+4) \& 0xaa) >> 1) | (((0)+2) \& 0xa$ ) + 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)&0xaa)>> 1) | ((((0)+  $1) \mid ((((\ (\ 0\ )\ +\ 4\ )\ \&\ 0x55) << 1))\ \&\ 0x33) << 2))\ \&\ 0x0f) << 4)), (((((((((((((\ (\ 0\ )\ +\ 4\ )\ +\ 1\ )\ \&\ 0xaa) >> 1)\ |\ ((((\ (\ 0\ )\ +\ 4\ )\ +\ 1\ )\ \&\ 0xaa) >> 1)\ |\ (((\ (\ (\ 0\ )\ +\ 4\ )\ +\ 1\ )\ \&\ 0xaa) >> 1)\ |\ (((\ (\ (\ 0\ )\ +\ 4\ )\ +\ 1\ )\ \&\ 0xaa) >> 1)\ |\ (((\ (\ (\ 0\ )\ +\ 4\ )\ +\ 1\ )\ \&\ 0xaa) >> 1)\ |\ (((\ (\ (\ 0\ )\ +\ 4\ )\ +\ 1\ )\ \&\ 0xaa) >> 1)\ |\ (((\ (\ (\ 0\ )\ +\ 4\ )\ +\ 1\ )\ \&\ 0xaa) >> 1)\ |\ (((\ (\ (\ 0\ )\ +\ 4\ )\ +\ 1\ )\ \&\ 0xaa) >> 1)\ |\ (((\ (\ (\ 0\ )\ +\ 4\ )\ +\ 1\ )\ \&\ 0xaa) >> 1)\ |\ (((\ (\ (\ 0\ )\ +\ 4\ )\ +\ 1\ )\ \&\ 0xaa) >> 1)\ |\ (((\ (\ (\ 0\ )\ +\ 4\ )\ +\ 1\ )\ \&\ 0xaa) >> 1)\ |\ (((\ (\ (\ 0\ )\ +\ 4\ )\ +\ 1\ )\ \&\ 0xaa) >> 1)\ |\ (((\ (\ (\ 0\ )\ +\ 4\ )\ +\ 1\ )\ \&\ 0xaa) >> 1)\ |\ (((\ (\ (\ 0\ )\ +\ 4\ )\ +\ 1\ )\ \&\ 0xaa) >> 1)\ |\ (((\ (\ (\ 0\ )\ +\ 4\ )\ +\ 1\ )\ \&\ 0xaa) >> 1)\ |\ (((\ (\ (\ 0\ )\ +\ 4\ )\ +\ 1\ )\ \&\ 0xaa) >> 1)\ |\ (((\ (\ (\ 0\ )\ +\ 4\ )\ +\ 1\ )\ \&\ 0xaa) >> 1)\ |\ (((\ (\ (\ 0\ )\ +\ 4\ )\ +\ 1\ )\ \&\ 0xaa) >> 1)\ |\ (((\ (\ (\ 0\ )\ +\ 4\ )\ +\ 1\ )\ \&\ 0xaa) >> 1)\ |\ (((\ (\ (\ 0\ )\ +\ 4\ )\ +\ 1\ )\ \&\ 0xaa) >> 1)\ |\ (((\ (\ (\ 0\ )\ +\ 4\ )\ +\ 1\ )\ \&\ 0xaa) >> 1)\ |\ (((\ (\ (\ 0\ )\ +\ 4\ )\ +\ 1\ )\ \&\ 0xaa) >> 1)\ |\ (((\ (\ 0\ )\ +\ 4\ )\ +\ 1\ )\ \&\ 0xaa) >> 1)\ |\ (((\ (\ 0\ )\ +\ 4\ )\ +\ 1\ )\ \&\ 0xaa) >> 1)\ |\ (((\ (\ 0\ )\ +\ 4\ )\ +\ 1\ )\ \&\ 0xaa) >> 1)\ |\ (((\ (\ 0\ )\ +\ 4\ )\ +\ 1\ )\ \&\ 0xaa) >> 1)\ |\ (((\ (\ 0\ )\ +\ 4\ )\ +\ 1\ )\ \&\ 0xaa) >> 1)\ |\ (((\ (\ 0\ )\ +\ 4\ )\ +\ 1\ )\ \&\ 0xaa) >> 1)\ |\ (((\ 0\ )\ +\ 4\ )\ +\ 1\ )\ \&\ 0xaa) >> 1)\ |\ (((\ 0\ )\ +\ 1)\ +\ 1\ )\ \&\ 0xaa) >> 1)\ |\ ((\ 0\ )\ +\ 1)\ ((\ 0\ )\ +\ 1)\ )\ ((\ 0\ )\ )\ )$  |\ ((\ 0\ )\ )\ ((\ 0\ )\ )\ ) |\ ((\ 0\ )\ )\ ((\ 0\ )\ )\ ) |\ ((\ 0\ )\ )\ ((\ 0\ )\ )\ ) |\ ((\ 0\ )\ )\ ((\ 0\ )\ )\ )\ ((\ 0\ )\ )\ ((\ 0\ )\ )\ ((\ 0\ )\ )\ ) |\ ((\ 0\ )\ )\ ((\ 0\ )\ )\ )\ ((\ 0\ )\ )\ ((\ 0\ )\ )\ ((\ 0\ )\ )\ )\ ((\ 0\ )\ )\ ((\ 0\ )\ )\ ((\ 0\ )\ )\ ((\ 0\ )\ )\ ((\ 0\ )\ )\ ((\ 0\ )\  $) \& 0x55) <<1)) \& 0xcc)>>2) \mid (((((((0)+4)+1) \& 0xaa)>>1) \mid (((((0)+4)+1) \& 0x55) <<1)) \& 0x33) <<1)$ 1)) & 0xcc) >> 2) | (((((((0)+4)+2) & 0xaa) >> 1) | (((((0)+4)+2) & 0x5) << 1)) & 0x33) << 2)) & 0x0f) <<)+2)+1) & 0xaa) >> 1) | (((((0)+4)+2)+1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((0)+4)+2)+1) & 0xaa) ) + 8 ) & 0x55) << 1)) & 0xcc) >> 2) | (((((( ( 0 ) + 8 ) & 0xaa) >> 1) | ((( ( 0 ) + 8 ) & 0x55) << 1)) & 0x33) << 2)) & ) & 0x55) << 1)) & 0xcc) >> 2) | (((((((0)+8)+1)&0xaa)>>1) | (((((0)+8)+1)&0x55)<<1)) & 0x33) <<+8) +1) & 0xaa) >>1) | (((((0) +8) +1) & 0x55) <<1)) & 0x33) <<2)) & 0x0f) <<4), ((((((((((0) +8) +2) & (10) +2 1)) & 0xcc) >> 2) | (((((((0)+8)+2) & 0xaa) >> 1) | (((((0)+8)+2) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 1>> 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) & 0xaa >> 1 | (((((0)+8)+4)+1) & 0x55 >< 1) & 0xcc >> 2 | (((((((0)+8)+4)+1) & 0xaa >> 1) | (((( << 1)) & 0xcc) >> 2) | ((((((((0)+8)+4)+2)+1) & 0xaa) >> 1) | ((((((0)+8)+4)+2)+1) & 0x55) << 1)) << 1)) & 0x33) << 2)) & 0x0f) << 4)), ((((((((( ( 0 ) + 16 ) & 0xaa) >> 1) | (((( ( 0 ) + 16 ) & 0x55) << 1)) & 0xcc) >> 2)  $2) \mid (((((((0)+16)+1) \& 0xaa) >> 1) \mid (((((0)+16)+1) \& 0x55) << 1)) \& 0x33) << 2)) \& 0xf0) >> 4) \mid (((((((0)+16)+1) \& 0x55) << 1)) \& 0x33) << 2)) \& 0xf0) >> 4) \mid ((((((((0)+16)+1) \& 0x55) << 1)) \& 0x55) << 1)) \& 0x55) << 1) & 0x55 << 1)$  $\frac{16) + 2) & 0x55}{((((((0) + 16) + 2) & 0xaa) >> 1) | (((((0) + 16) + 2) & 0x55) << 1)) & -2) & 0x55) << 1) | ((((0) + 16) + 2) & 0x55) << 1) | (((0) + 16) + 2) & 0x55) << 1) | (((0) + 16) + 2) & 0x55) << 1) | (((0) + 16) + 2) & 0x55) << 1) | (((0) + 16) + 2) & 0x55) << 1) | (((0) + 16) + 2) & 0x55) << 1) | (((0) + 16) + 2) & 0x55) << 1) | (((0) + 16) + 2) & 0x55) << 1) | (((0) + 16) + 2) & 0x55) << 1) | (((0) + 16) + 2) & 0x55) << 1) | (((0) + 16) + 2) & 0x55) << 1) | (((0) + 16) + 2) & 0x55) << 1) | ((0) + 16) + 2) & 0x55) << 1) | ((0) + 16) + 2) & 0x55) << 1) | ((0) + 16) + 2) & 0x55) << 1) | ((0) + 16) + 2) & 0x55) << 1) | ((0) + 16) + 2) & 0x55) << 1) | ((0) + 16) + 2) & 0x55) << 1) | (0) + 16) + 2) & 0x55) << 1) | (0) + 16) + 2) & 0x55) << 1) | (0) + 16) + 2) & 0x55) << 1) | (0) + 16) + 2) & 0x55) << 1) | (0) + 16) + 2) & 0x55) << 1) | (0) + 16) + 2) & 0x55$ Generated  $\frac{1}{2}$  Gener (0) + 16 + 2 + 1 & 0 (0) + 2 + 1 & 0 (0) + 2 + 1 & 0 (0) + 2 + 1 & 0 (0) + 2 + 1 & 0 (0) + 2 + 1 & 0 (0) + 2 & 0

Referenced by ordered8to3().

## 8.31 src/ckb-daemon/led\_mouse.c File Reference

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

Include dependency graph for led\_mouse.c:



### **Functions**

- static int rgbcmp (const lighting \*lhs, const lighting \*rhs)
- static int isblack (const usbdevice \*kb, const lighting \*light)
- int updatergb\_mouse (usbdevice \*kb, int force)
- int savergb\_mouse (usbdevice \*kb, lighting \*light, int mode)
- int loadrgb\_mouse (usbdevice \*kb, lighting \*light, int mode)

#### 8.31.1 Function Documentation

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

Definition at line 13 of file led mouse.c.

 $References\ lighting::b,\ lighting::g,\ IS\_M65,\ LED\_MOUSE,\ N\_MOUSE\_ZONES,\ and\ lighting::r.$ 

Referenced by updatergb\_mouse().



8.31.1.2 int loadrgb\_mouse ( usbdevice \* kb, lighting \* light, int mode )

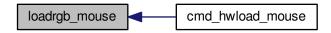
Definition at line 81 of file led\_mouse.c.

References lighting::b, ckb\_err, lighting::g, IS\_SABRE, IS\_SCIMITAR, LED\_DPI, LED\_MOUSE, MSG\_SIZE, lighting::r, and usbrecv.

Referenced by cmd\_hwload\_mouse().

```
82
        uchar data_pkt[MSG_SIZE] = { 0x0e, 0x13, 0x10, 1, 0 };
        uchar in_pkt[MSG_SIZE] = { 0 };
// Load each RGB zone
8.3
84
        int zonecount = IS_SCIMITAR(kb) ? 4 : IS_SABRE(kb) ? 3 : 2;
85
        for (int i = 0; i < zonecount; i++) {</pre>
            if(!usbrecv(kb, data_pkt, in_pkt))
88
                  return -1;
            if (memcmp(in_pkt, data_pkt, 4)) {
    ckb_err("Bad input header\n");
89
90
                 return -2;
91
            // Copy data
            int led = LED_MOUSE + i;
if(led >= LED_DPI)
95
                                    // Skip DPI light
                 led++;
96
            light->r[led] = in_pkt[4];
97
             light->g[led] = in_pkt[5];
99
            light->b[led] = in_pkt[6];
              // Set packet for next zone
100
101
              data_pkt[2]++;
103
         return 0;
104 }
```

Here is the caller graph for this function:



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

Definition at line 7 of file led\_mouse.c.

References lighting::b, lighting::g, LED\_MOUSE, N\_MOUSE\_ZONES, and lighting::r.

Referenced by updatergb\_mouse().

Here is the caller graph for this function:



8.31.1.4 int savergb\_mouse ( usbdevice \* kb, lighting \* light, int mode )

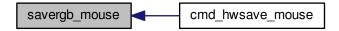
Definition at line 62 of file led\_mouse.c.

References lighting::b, lighting::g, IS\_SABRE, IS\_SCIMITAR, LED\_DPI, LED\_MOUSE, MSG\_SIZE, lighting::r, and usbsend.

Referenced by cmd\_hwsave\_mouse().

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

Here is the caller graph for this function:



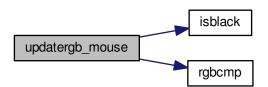
8.31.1.5 int updatergb\_mouse ( usbdevice \* kb, int force )

Definition at line 20 of file led\_mouse.c.

References usbdevice::active, lighting::b, usbprofile::currentmode, lighting::forceupdate, lighting::g, isblack(), usbprofile::lastlight, LED\_MOUSE, usbmode::light, MSG\_SIZE, N\_MOUSE\_ZONES, usbdevice::profile, lighting::r, rgbcmp(), and usbsend.

```
20
2.1
       if(!kb->active)
2.2
           return 0;
       lighting* lastlight = &kb->profile->lastlight;
23
       lighting* newlight = &kb->profile->currentmode->
25
      // Don't do anything if the lighting hasn't changed
26
       if(!force && !lastlight->forceupdate && !newlight->forceupdate
           && !rgbcmp(lastlight, newlight))
return 0;
27
28
      lastlight->forceupdate = newlight->forceupdate = 0;
29
30
      \ensuremath{//} Send the RGB values for each zone to the mouse
32
33
34
35
36
       uchar* rgb_data = &data_pkt[0][4];
37
       for(int i = 0; i < N_MOUSE_ZONES; i++) {</pre>
           *rgb_data++ = i + 1;
*rgb_data++ = newlight->r[LED_MOUSE + i];
*rgb_data++ = newlight->g[LED_MOUSE + i];
38
39
40
           *rgb_data++ = newlight->b[LED_MOUSE + i];
41
42
43
       // Send RGB data
44
       if(!usbsend(kb, data_pkt[0], 1))
4.5
           return -1;
       int was_black = isblack(kb, lastlight), is_black = isblack(kb, newlight);
46
47
       if(is black){
           // If the lighting is black, send the deactivation packet (M65 only)
48
49
           if(!usbsend(kb, data_pkt[1], 1))
50
               return -1;
      } else if(was_black || force){
51
          // If the lighting WAS black, or if we're on forced update, send the activation packet
52
53
           data pkt[1][4] = 1;
54
           if(!usbsend(kb, data_pkt[1], 1))
               return -1;
57
58
       memcpy(lastlight, newlight, sizeof(lighting));
59
       return 0:
60 }
```

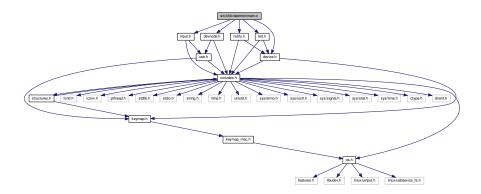
Here is the call graph for this function:



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

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

Include dependency graph for main.c:



## **Functions**

- static void quitWithLock (char mut) quitWithLock
- int restart ()
- void timespec\_add (struct timespec \*timespec, long nanoseconds)
- static void quit ()

quit Stop working the daemon. function is called if the daemon received a sigterm In this case, locking the device-mutex is ok.

- void sighandler2 (int type)
- void sighandler (int type)
- void localecase (char \*dst, size\_t length, const char \*src)
- int main (int argc, char \*\*argv)

## **Variables**

- static int main\_ac
- static char \*\* main\_av
- volatile int reset stop
- · int features\_mask
- int hwload\_mode

# 8.32.1 Function Documentation

8.32.1.1 void localecase ( char \* dst, size\_t length, const char \* src )

Definition at line 71 of file main.c.

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

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

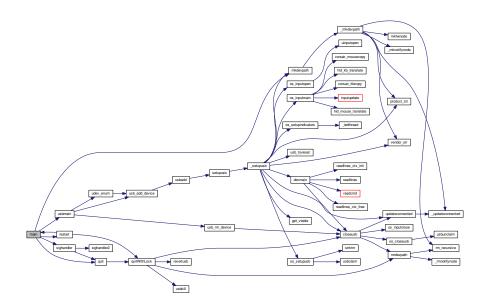
Definition at line 88 of file main.c.

References ckb\_fatal\_nofile, ckb\_info, ckb\_info\_nofile, ckb\_warn\_nofile, devpath, FEAT\_BIND, FEAT\_MOUSE-ACCEL, FEAT\_NOTIFY, features\_mask, gid, hwload\_mode, keyboard, main\_ac, main\_av, mkdevpath(), quit(), restart(), sighandler(), and usbmain().

Referenced by restart().

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

```
154
              char* argument = argv[i];
155
              unsigned newgid;
156
              char hwload[7];
              if(sscanf(argument, "--gid=%u", &newgid) == 1){
157
                  // Set dev node GID
158
159
                   gid = newgid;
              ckb_info_nofile("Setting /dev node gid: %u\n", newgid);
} else if(!strcmp(argument, "--nobind")){
160
161
                   // Disable key notifications and rebinding
162
163
                   features_mask &= ~FEAT_BIND & ~FEAT_NOTIFY;
              ckb_info_nofile("Key binding and key notifications are disabled\n");
} else if(!strcmp(argument, "--nonotify"));
164
165
                   // Disable key notifications
features_mask &= ~FEAT_NOTIFY;
166
167
168
                   ckb\_info\_nofile("Key notifications are disabled\n");
              } else if(sscanf(argument, "--hwload=%6s", hwload) == 1) {
   if(!strcmp(hwload, "always") || !strcmp(hwload, "yes") || !strcmp(hwload, "y") || !strcmp(
169
170
       hwload, "a")){
171
                       hwload_mode = 2;
                       ckb_info_nofile("Setting hardware load: always\n");
       } else if(!strcmp(hwload, "tryonce") || !strcmp(hwload, "try") || !strcmp(hwload, "once") || !strcmp(hwload, "t") || !strcmp(hwload, "o")){
173
                   hwload_mode = 1;
  ckb_info_nofile("Setting hardware load: tryonce\n");
} else if(!strcmp(hwload, "never") || !strcmp(hwload, "none") || !strcmp(hwload, "no") || !
174
175
176
       strcmp(hwload, "n")){
177
                       hwload_mode = 0;
178
                       ckb_info_nofile("Setting hardware load: never\n");
179
180
              } else if(!strcmp(argument, "--nonroot")){
                  // Allow running as a non-root user
181
182
                   forceroot = 0;
183
184 #ifdef OS_MAC
185
              else if(!strcmp(argument, "--nomouseaccel")){
                  // On OSX, provide an option to disable mouse acceleration
features_mask &= ~FEAT_MOUSEACCEL;
186
187
                   ckb_info_nofile("Mouse acceleration disabled\n");
188
189
              }
190 #endif
191
192
         // Check UID
193
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))
              ckb_info("Root controller ready at %s0\n", devpath);
206
207
208
         // Set signals
209
         sigset_t signals;
210
         sigfillset(&signals);
         sigdelset(&signals, SIGTERM);
sigdelset(&signals, SIGINT);
211
212
         sigdelset(&signals, SIGQUIT);
213
214
         sigdelset(&signals, SIGUSR1);
215
         // Set up signal handlers for quitting the service.
216
         sigprocmask(SIG_SETMASK, &signals, 0);
         signal(SIGTERM, sighandler);
217
         signal(SIGINT, sighandler);
signal(SIGQUIT, sighandler);
218
219
220
         signal(SIGUSR1, (void (*)())restart);
221
222
         // Start the USB system
223
         int result = usbmain();
224
         quit();
225
         return result;
226 }
```



Here is the caller graph for this function:



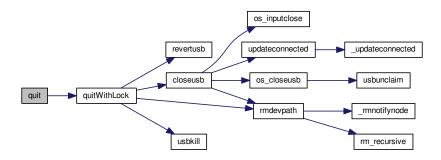
```
8.32.1.3 static void quit ( ) [static]
```

Definition at line 30 of file main.c.

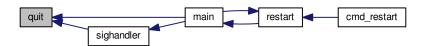
References quitWithLock().

Referenced by main(), and sighandler().

Here is the call graph for this function:



Here is the caller graph for this function:



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

#### **Parameters**

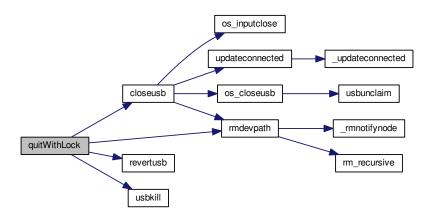
```
mut | try to close files maybe without locking the mutex if mut == true then lock
```

Definition at line 40 of file main.c.

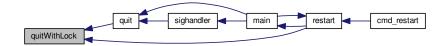
References ckb\_info, closeusb(), DEV\_MAX, devmutex, IS\_CONNECTED, keyboard, reset\_stop, revertusb(), rmdevpath(), and usbkill().

Referenced by quit(), and restart().

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



Here is the caller graph for this function:



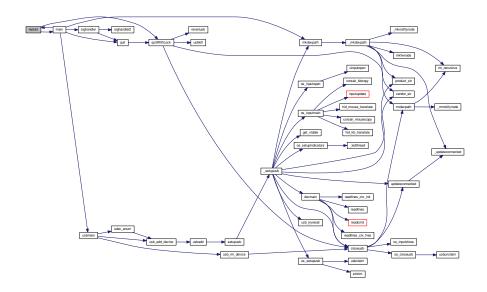
```
8.32.1.5 int restart ( )
```

Definition at line 228 of file main.c.

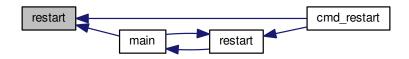
References ckb\_err, main(), main\_ac, main\_av, and quitWithLock().

Referenced by cmd\_restart(), and main().

Here is the call graph for this function:



Here is the caller graph for this function:

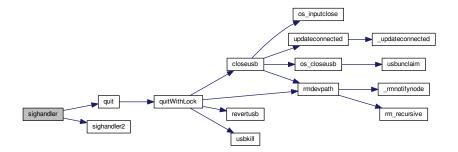


# 8.32.1.6 void sighandler ( int type )

Definition at line 62 of file main.c.

References quit(), and sighandler2().

Referenced by main().



Here is the caller graph for this function:



## 8.32.1.7 void sighandler2 (int type)

Definition at line 57 of file main.c.

Referenced by sighandler().

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

Here is the caller graph for this function:



## 8.32.1.8 void timespec\_add ( struct timespec \* timespec, long nanoseconds )

Definition at line 19 of file main.c.

## 8.32.2 Variable Documentation

8.32.2.1 int features\_mask

Definition at line 17 of file usb.c.

Referenced by \_setupusb(), and main().

8.32.2.2 int hwload\_mode

Definition at line 7 of file device.c.

Referenced by main().

```
8.32.2.3 int main_ac [static]
```

Definition at line 7 of file main.c.

Referenced by main(), and restart().

```
8.32.2.4 char** main_av [static]
```

Definition at line 8 of file main.c.

Referenced by main(), and restart().

8.32.2.5 volatile int reset\_stop

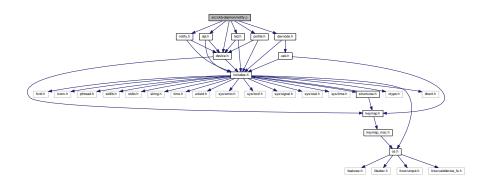
Definition at line 14 of file usb.c.

Referenced by \_usbrecv(), \_usbsend(), quitWithLock(), and usb\_tryreset().

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

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

Include dependency graph for notify.c:



#### **Macros**

- #define HWMODE OR RETURN(kb, index)
- #define HW\_STANDARD

#### **Functions**

- void nprintf (usbdevice \*kb, int nodenumber, usbmode \*mode, const char \*format....)
- void nprintkey (usbdevice \*kb, int nnumber, int keyindex, int down)
- void nprintind (usbdevice \*kb, int nnumber, int led, int on)
- void cmd\_notify (usbdevice \*kb, usbmode \*mode, int nnumber, int keyindex, const char \*toggle)
- static void \_cmd\_get (usbdevice \*kb, usbmode \*mode, int nnumber, const char \*setting)
- void cmd\_get (usbdevice \*kb, usbmode \*mode, int nnumber, int dummy, const char \*setting)
- int restart ()
- void cmd\_restart (usbdevice \*kb, usbmode \*mode, int nnumber, int dummy, const char \*content)

#### 8.33.1 Macro Definition Documentation

#### 8.33.1.1 #define HW\_STANDARD

#### Value:

Definition at line 83 of file notify.c.

Referenced by \_cmd\_get().

#### 8.33.1.2 #define HWMODE\_OR\_RETURN( kb, index )

## Value:

Definition at line 73 of file notify.c.

#### 8.33.2 Function Documentation

```
8.33.2.1 static void \_cmd\_get ( usbdevice * kb, usbmode * mode, int nnumber, const char * setting ) [static]
```

Definition at line 90 of file notify.c.

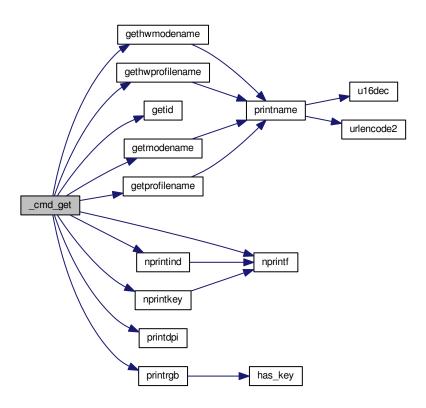
References dpiset::current, usbmode::dpi, hwprofile::dpi, gethwmodename(), gethwprofilename(), getid(), getmodename(), getprofilename(), usbdevice::hw, usbdevice::hw\_ileds, HW\_STANDARD, I\_CAPS, I\_NUM, I\_SCROLL, usbmode::id, usbprofile::id, hwprofile::id, usbdevice::input, keymap, usbinput::keys, dpiset::lift, usbmode::light, hwprofile::light, usbid::modified, N\_KEYS\_INPUT, nprintf(), nprintind(), nprintkey(), printdpi(), printrgb(), usbdevice::profile, and dpiset::snap.

Referenced by cmd\_get().

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

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

Here is the call graph for this function:



Here is the caller graph for this function:

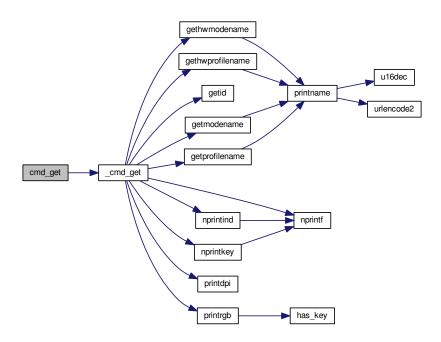


8.33.2.2 void cmd\_get ( usbdevice \* kb, usbmode \* mode, int nnumber, int dummy, const char \* setting )

Definition at line 215 of file notify.c.

References \_cmd\_get(), and imutex.

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



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

Definition at line 61 of file notify.c.

References CLEAR\_KEYBIT, imutex, N\_KEYS\_INPUT, usbmode::notify, and SET\_KEYBIT.

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

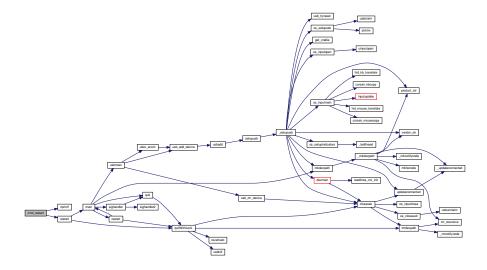
8.33.2.4 void cmd\_restart ( usbdevice \* kb, usbmode \* mode, int nnumber, int dummy, const char \* content )

Definition at line 223 of file notify.c.

References ckb\_info, nprintf(), and restart().

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

Here is the call graph for this function:



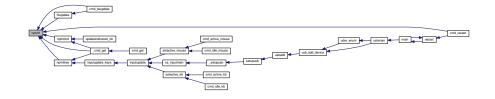
8.33.2.5 void nprintf ( usbdevice \* kb, int nodenumber, usbmode \* mode, const char \* format, ... )

Definition at line 8 of file notify.c.

References INDEX\_OF, usbprofile::mode, usbdevice::outfifo, OUTFIFO\_MAX, and usbdevice::profile. Referenced by \_cmd\_get(), cmd\_fwupdate(), cmd\_restart(), fwupdate(), nprintind(), and nprintkey().

```
8
       if(!kb)
10
            return;
11
        usbprofile* profile = kb->profile;
        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){
                 va_start(va_args, format);
18
                 if (mode)
                     dprintf(fifo, "mode %d ", INDEX_OF(mode, profile->mode) + 1);
19
                 vdprintf(fifo, format, va_args);
2.0
21
22
            return;
        // Otherwise, print to all nodes
for(int i = 0; i < OUTFIFO_MAX; i++) {
    if((fifo = kb->outfifo[i] - 1) != -1) {
24
2.5
26
27
                 va_start(va_args, format);
28
                 if (mode)
29
                     dprintf(fifo, "mode %d ", INDEX_OF(mode, profile->mode) + 1);
30
                 vdprintf(fifo, format, va_args);
31
32
33 }
```

Here is the caller graph for this function:



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

Definition at line 43 of file notify.c.

References I\_CAPS, I\_NUM, I\_SCROLL, and nprintf().

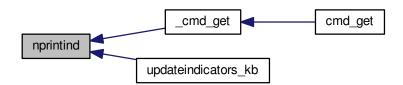
Referenced by \_cmd\_get(), and updateindicators\_kb().

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

Here is the call graph for this function:



Here is the caller graph for this function:



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

Definition at line 35 of file notify.c.

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

Referenced by \_cmd\_get(), and inputupdate\_keys().

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

Here is the call graph for this function:



Here is the caller graph for this function:

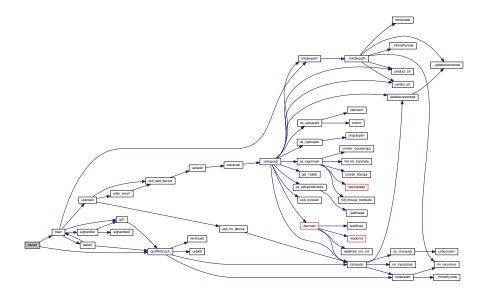
```
Total State No. 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 - 1900 -
```

```
8.33.2.8 int restart ( )
```

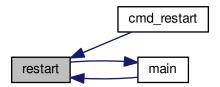
Definition at line 228 of file main.c.

References ckb\_err, main(), main\_ac, main\_av, and quitWithLock().

Referenced by cmd\_restart(), and main().

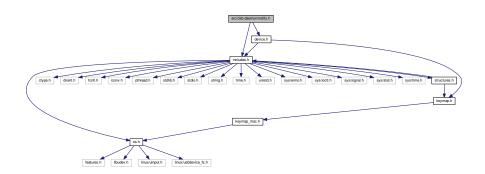


Here is the caller graph for this function:



# 8.34 src/ckb-daemon/notify.h File Reference

#include "includes.h"
#include "device.h"
Include dependency graph for notify.h:



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



## **Functions**

- void nprintf (usbdevice \*kb, int nodenumber, usbmode \*mode, const char \*format,...)
- void nprintkey (usbdevice \*kb, int nnumber, int keyindex, int down)
- void nprintind (usbdevice \*kb, int nnumber, int led, int on)
- void cmd\_notify (usbdevice \*kb, usbmode \*mode, int nnumber, int keyindex, const char \*toggle)
- void cmd\_get (usbdevice \*kb, usbmode \*mode, int nnumber, int dummy, const char \*setting)
- · void cmd\_restart (usbdevice \*kb, usbmode \*mode, int nnumber, int dummy, const char \*content)

#### 8.34.1 Function Documentation

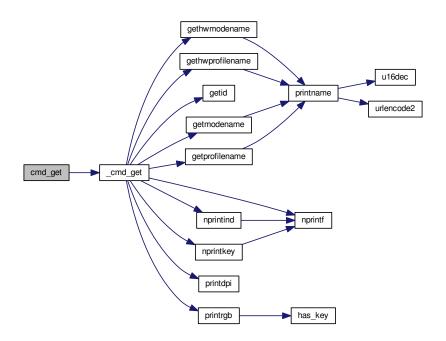
8.34.1.1 void cmd\_get ( usbdevice \* kb, usbmode \* mode, int nnumber, int dummy, const char \* setting )

Definition at line 215 of file notify.c.

References \_cmd\_get(), and imutex.

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

Here is the call graph for this function:



8.34.1.2 void cmd\_notify ( usbdevice \* kb, usbmode \* mode, int nnumber, int keyindex, const char \* toggle )

Definition at line 61 of file notify.c.

References CLEAR KEYBIT, imutex, N KEYS INPUT, usbmode::notify, and SET KEYBIT.

```
61
62
       if(keyindex >= N_KEYS_INPUT)
63
           return;
       pthread_mutex_lock(imutex(kb));
64
       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);
       pthread_mutex_unlock(imutex(kb));
69
70 }
```

8.34.1.3 void cmd\_restart ( usbdevice \* kb, usbmode \* mode, int nnumber, int dummy, const char \* content )

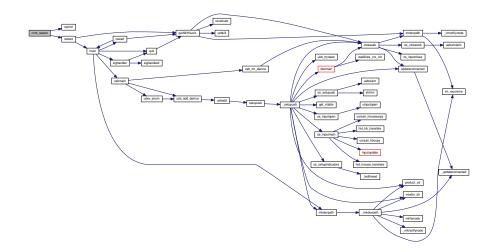
Definition at line 223 of file notify.c.

References ckb\_info, nprintf(), and restart().

```
223

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

Here is the call graph for this function:



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

Definition at line 8 of file notify.c.

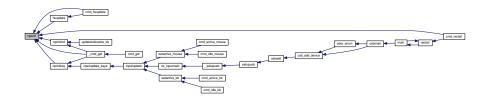
References INDEX\_OF, usbprofile::mode, usbdevice::outfifo, OUTFIFO\_MAX, and usbdevice::profile.

Referenced by \_cmd\_get(), cmd\_fwupdate(), cmd\_restart(), fwupdate(), nprintind(), and nprintkey().

```
8
9    if(!kb)
10        return;
11    usbprofile* profile = kb->profile;
12    va_list va_args;
13    int fifo;
```

```
14
        if(nodenumber >= 0){
            // 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)
                      dprintf(fifo, "mode %d ", INDEX_OF(mode, profile->mode) + 1);
19
                 vdprintf(fifo, format, va_args);
20
21
22
             return;
23
        // Otherwise, print to all nodes
for(int i = 0; i < OUTFIFO_MAX; i++) {
    if((fifo = kb->outfifo[i] - 1) != -1) {
24
25
26
27
                 va_start(va_args, format);
28
                 if (mode)
29
                      dprintf(fifo, "mode %d ", INDEX_OF(mode, profile->mode) + 1);
30
                 vdprintf(fifo, format, va_args);
31
            }
32
        }
33 }
```

Here is the caller graph for this function:



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

Definition at line 43 of file notify.c.

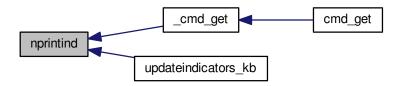
References I CAPS, I NUM, I SCROLL, and nprintf().

Referenced by \_cmd\_get(), and updateindicators\_kb().

```
44
       const char* name = 0;
4.5
       switch(led){
46
       case I NUM:
          name = "num";
47
48
          break;
49
       case I_CAPS:
50
         name = "caps";
51
           break;
      case I_SCROLL:
52
         name = "scroll";
53
           break;
55
       default:
56
           return;
57
      nprintf(kb, nnumber, 0, "i %c%s\n", on ? '+' : '-', name);
58
59 }
```

Here is the call graph for this function:





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

Definition at line 35 of file notify.c.

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

Referenced by \_cmd\_get(), and inputupdate\_keys().

Here is the call graph for this function:



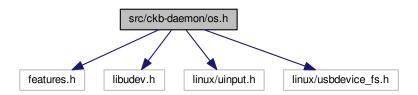
Here is the caller graph for this function:



# 8.35 src/ckb-daemon/os.h File Reference

#include <features.h>

```
#include <libudev.h>
#include <linux/uinput.h>
#include <linux/usbdevice_fs.h>
Include dependency graph for os.h:
```



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



#### **Macros**

- #define \_DEFAULT\_SOURCE
- #define GNU SOURCE
- #define UINPUT\_VERSION 2
- #define euid\_guard\_start
- #define euid\_guard\_stop

#### 8.35.1 Macro Definition Documentation

8.35.1.1 #define \_DEFAULT\_SOURCE

Definition at line 22 of file os.h.

8.35.1.2 #define \_GNU\_SOURCE

Definition at line 26 of file os.h.

8.35.1.3 #define euid\_guard\_start

Definition at line 40 of file os.h.

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

8.35.1.4 #define euid\_guard\_stop

Definition at line 41 of file os.h.

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

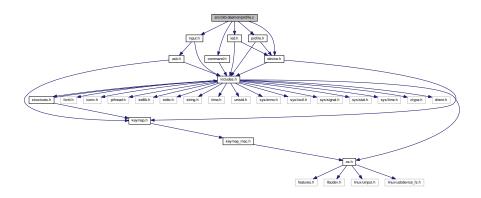
#### 8.35.1.5 #define UINPUT\_VERSION 2

Definition at line 35 of file os.h.

## 8.36 src/ckb-daemon/profile.c File Reference

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

Include dependency graph for profile.c:



## **Functions**

- void urldecode2 (char \*dst, const char \*src)
- void urlencode2 (char \*dst, const char \*src)
- int setid (usbid \*id, const char \*guid)
- char \* getid (usbid \*id)
- void u16enc (char \*in, ushort \*out, size t \*srclen, size t \*dstlen)
- void u16dec (ushort \*in, char \*out, size\_t \*srclen, size\_t \*dstlen)
- void cmd\_name (usbdevice \*kb, usbmode \*mode, int dummy1, int dummy2, const char \*name)
- void cmd\_profilename (usbdevice \*kb, usbmode \*dummy1, int dummy2, int dummy3, const char \*name)
- char \* printname (ushort \*name, int length)
- char \* getmodename (usbmode \*mode)
- char \* getprofilename (usbprofile \*profile)
- char \* gethwmodename (hwprofile \*profile, int index)
- char \* gethwprofilename (hwprofile \*profile)
- void cmd\_id (usbdevice \*kb, usbmode \*mode, int dummy1, int dummy2, const char \*id)
- void cmd\_profileid (usbdevice \*kb, usbmode \*mode, int dummy1, int dummy2, const char \*id)
- static void initmode (usbmode \*mode)
- void allocprofile (usbdevice \*kb)
- int loadprofile (usbdevice \*kb)
- static void freemode (usbmode \*mode)
- void cmd\_erase (usbdevice \*kb, usbmode \*mode, int dummy1, int dummy2, const char \*dummy3)
- static void \_freeprofile (usbdevice \*kb)
- void cmd eraseprofile (usbdevice \*kb, usbmode \*dummy1, int dummy2, int dummy3, const char \*dummy4)
- void freeprofile (usbdevice \*kb)
- void hwtonative (usbprofile \*profile, hwprofile \*hw, int modecount)
- void nativetohw (usbprofile \*profile, hwprofile \*hw, int modecount)

## **Variables**

```
static iconv_t utf8to16 = 0static iconv_t utf16to8 = 0
```

## 8.36.1 Function Documentation

```
8.36.1.1 static void _freeprofile ( usbdevice * kb ) [static]
```

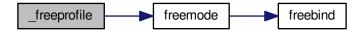
Definition at line 210 of file profile.c.

References freemode(), usbprofile::mode, MODE\_COUNT, and usbdevice::profile.

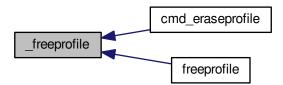
Referenced by cmd\_eraseprofile(), and freeprofile().

```
210
211    usbprofile* profile = kb->profile;
212    if(!profile)
213        return;
214    // Clear all mode data
215    for(int i = 0; i < MODE_COUNT; i++)
216        freemode(profile->mode + i);
217    free(profile);
218    kb->profile = 0;
219 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



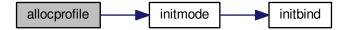
## 8.36.1.2 void allocprofile ( usbdevice \* kb )

Definition at line 182 of file profile.c.

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

Referenced by cmd\_eraseprofile().

Here is the call graph for this function:



Here is the caller graph for this function:



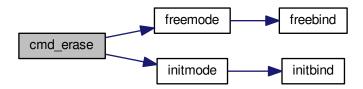
8.36.1.3 void cmd\_erase ( usbdevice \* kb, usbmode \* mode, int dummy1, int dummy2, const char \* dummy3 )

Definition at line 203 of file profile.c.

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

```
203
204 pthread_mutex_lock(imutex(kb));
205 freemode(mode);
206 initmode(mode);
207 pthread_mutex_unlock(imutex(kb));
208 }
```

Here is the call graph for this function:



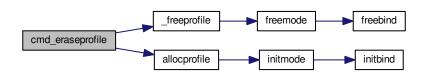
8.36.1.4 void cmd\_eraseprofile ( usbdevice \* kb, usbmode \* dummy1, int dummy2, int dummy3, const char \* dummy4 )

Definition at line 221 of file profile.c.

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

```
221
222    pthread_mutex_lock(imutex(kb));
223    _freeprofile(kb);
224    allocprofile(kb);
225    pthread_mutex_unlock(imutex(kb));
226 }
```

Here is the call graph for this function:



8.36.1.5 void cmd\_id ( usbdevice \* kb, usbmode \* mode, int dummy1, int dummy2, const char \* id )

Definition at line 160 of file profile.c.

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

```
160

161  // ID is either a GUID or an 8-digit hex number

162  int newmodified;

163  if(!setid(&mode->id, id) && sscanf(id, "%08x", &newmodified) == 1)

164  memcpy(mode->id.modified, &newmodified, sizeof(newmodified));

165 }
```

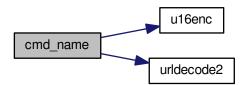


8.36.1.6 void cmd\_name ( usbdevice \* kb, usbmode \* mode, int dummy1, int dummy2, const char \* name )

Definition at line 117 of file profile.c.

References MD\_NAME\_LEN, usbmode::name, u16enc(), and urldecode2().

Here is the call graph for this function:



8.36.1.7 void cmd\_profileid ( usbdevice \* kb, usbmode \* mode, int dummy1, int dummy2, const char \* id )

Definition at line 167 of file profile.c.

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

```
167

168 usbprofile* profile = kb->profile;

169 int newmodified;

170 if(!setid(&profile->id, id) && sscanf(id, "%08x", &newmodified) == 1)

171 memcpy(profile->id.modified, &newmodified, sizeof(newmodified));

172

173 }
```

Here is the call graph for this function:



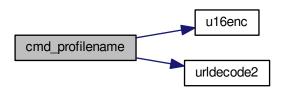
8.36.1.8 void cmd\_profilename ( usbdevice \* kb, usbmode \* dummy1, int dummy2, int dummy3, const char \* name )

Definition at line 124 of file profile.c.

References usbprofile::name, PR\_NAME\_LEN, usbdevice::profile, u16enc(), and urldecode2().

```
124
125    usbprofile* profile = kb->profile;
126    char decoded[strlen(name) + 1];
127    urldecode2(decoded, name);
128    size_t srclen = strlen(decoded), dstlen = PR_NAME_LEN;
129    ul6enc(decoded, profile->name, &srclen, &dstlen);
130 }
```

Here is the call graph for this function:



```
8.36.1.9 static void freemode ( usbmode * mode ) [static]
```

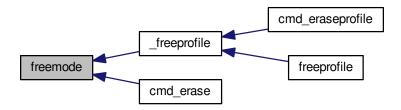
Definition at line 198 of file profile.c.

References usbmode::bind, and freebind().

Referenced by \_freeprofile(), and cmd\_erase().



Here is the caller graph for this function:

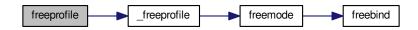


## 8.36.1.10 void freeprofile ( usbdevice \* kb )

Definition at line 228 of file profile.c.

References \_freeprofile(), and usbdevice::hw.

Here is the call graph for this function:



8.36.1.11 char\* gethwmodename ( hwprofile \* profile, int index )

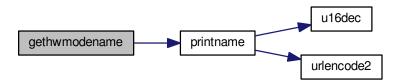
Definition at line 152 of file profile.c.

References MD\_NAME\_LEN, hwprofile::name, and printname().

Referenced by \_cmd\_get().

```
152 {
153     return printname(profile->name[index + 1], MD_NAME_LEN);
154 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



```
8.36.1.12 char* gethwprofilename ( hwprofile * profile )
```

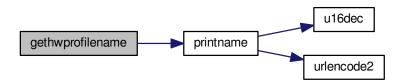
Definition at line 156 of file profile.c.

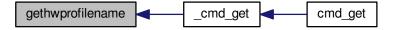
References MD\_NAME\_LEN, hwprofile::name, and printname().

Referenced by \_cmd\_get().

```
156
157     return printname(profile->name[0], MD_NAME_LEN);
158 }
```

Here is the call graph for this function:





```
8.36.1.13 char* getid ( usbid * id )
```

Definition at line 79 of file profile.c.

References usbid::guid.

Referenced by \_cmd\_get().

```
79
80
                                           int32_t data1;
                                           int16_t data2, data3, data4a;
                                           char data4b[6];
83
                                           memcpy(&data1, id->guid + 0x0, 4);
                                          memcpy(&data2, id->guid + 0x4, 2);
memcpy(&data3, id->guid + 0x6, 2);
84
85
                                          memcpy(&data4a, id->guid + 0x8, 2);
memcpy(&data4b, id->guid + 0xA, 6);
86
                                          char* guid = malloc(39);
snprintf(guid, 39, "{%08X-%04hX-%04hX-%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02
89
                                                                                                    data1, data2, data3, data4a, data4b[0], data4b[1], data4b[2], data4b[3], data4b[4], data4b[5])
90
91
                                           return guid;
92 }
```

Here is the caller graph for this function:



```
8.36.1.14 char* getmodename ( usbmode * mode )
```

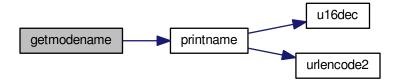
Definition at line 144 of file profile.c.

References MD NAME LEN, usbmode::name, and printname().

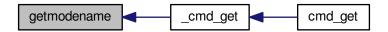
Referenced by \_cmd\_get().

```
144
145 return printname(mode->name, MD_NAME_LEN);
146 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



```
8.36.1.15 char* getprofilename ( usbprofile * profile )
```

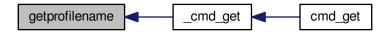
Definition at line 148 of file profile.c.

References usbprofile::name, PR\_NAME\_LEN, and printname().

Referenced by \_cmd\_get().

Here is the call graph for this function:





8.36.1.16 void hwtonative ( usbprofile \* profile, hwprofile \* hw, int modecount )

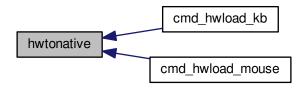
Definition at line 235 of file profile.c.

References usbmode::dpi, hwprofile::dpi, dpiset::forceupdate, lighting::forceupdate, usbmode::id, usbprofile::id, hwprofile::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().

```
236
         // Copy the profile name and ID
237
         memcpy(profile->name, hw->name[0], PR_NAME_LEN * 2);
238
         memcpy(&profile->id, hw->id, sizeof(usbid));
239
         // Copy the mode settings
         for (int i = 0; i < modecount; i++) {
240
241
             usbmode* mode = profile->mode + i;
             memcpy(mode->name, hw->name[i + 1], MD_NAME_LEN * 2);
242
243
             memcpy(&mode->id, hw->id + i + 1, sizeof(usbid));
244
             memcpy(&mode->light, hw->light + i, sizeof(lighting));
245
             memcpy(\&mode->dpi, hw->dpi + i, sizeof(dpiset));
             // Set a force update on the light/DPI since they've been overwritten mode->light.forceupdate = mode->dpi.forceupdate = 1;
246
247
248
249
        profile->lastlight.forceupdate = profile->lastdpi.
       forceupdate = 1;
250 }
```

Here is the caller graph for this function:



**8.36.1.17** static void initmode ( usbmode \* mode ) [static]

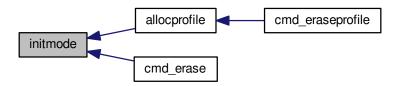
Definition at line 175 of file profile.c.

References usbmode::bind, usbmode::dpi, dpiset::forceupdate, lighting::forceupdate, initbind(), and usbmode::light. Referenced by allocprofile(), and cmd\_erase().

Here is the call graph for this function:



Here is the caller graph for this function:



## 8.36.1.18 int loadprofile ( usbdevice \* kb )

Definition at line 192 of file profile.c.

References hwloadprofile.

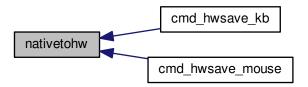
8.36.1.19 void nativetohw ( usbprofile \* profile, hwprofile \* hw, int modecount )

Definition at line 252 of file profile.c.

References usbmode::dpi, hwprofile::dpi, usbmode::id, usbprofile::id, hwprofile::id, usbmode::light, hwprofile::light, MD\_NAME\_LEN, usbprofile::mame, usbprofile::name, hwprofile::name, and PR\_NAME\_LEN.

Referenced by cmd\_hwsave\_kb(), and cmd\_hwsave\_mouse().

```
252 {
253    // Copy name and ID
254    memcpy(hw->name[0], profile->name, PR_NAME_LEN * 2);
255    memcpy(hw->id, &profile->id, sizeof(usbid));
256    // Copy the mode settings
```



```
8.36.1.20 char* printname ( ushort * name, int length )
```

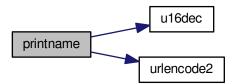
Definition at line 132 of file profile.c.

References u16dec(), and urlencode2().

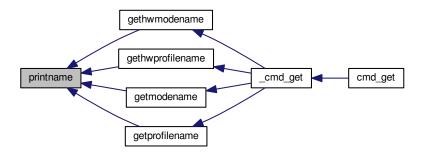
Referenced by gethwmodename(), gethwprofilename(), getmodename(), and getprofilename().

```
132
133
            // Convert the name to UTF-8
           char* buffer = calloc(1, length * 4 - 3);
size_t srclen = length, dstlen = length * 4 - 4;
134
135
           u16dec(name, buffer, &srclen, &dstlen);
// URL-encode it
char* buffer2 = malloc(strlen(buffer) * 3 + 1);
136
137
138
139
           urlencode2(buffer2, buffer);
140
           free(buffer);
141
           return buffer2;
142 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



8.36.1.21 int setid ( usbid \*id, const char \*guid )

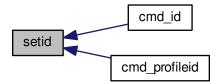
Definition at line 64 of file profile.c.

References usbid::guid.

Referenced by cmd\_id(), and cmd\_profileid().

```
64
65
                                         int32_t data1;
                                         int16_t data2, data3, data4a;
66
                                         char data4b[6];
                                          \textbf{if} (sscanf (guid, "\{\$08X-\$04hX-\$04hX-\$04hX-\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX\$02hhX$02hhX$02hhX$02hhX$02hhX$02hhX$02hhX$02hhX$02hhX$02hhX$02hhX$02hhX$02hhX$02hhX$02hhX$02hhX$02hhX$02hhX$02hhX$02hhX$02hhX$02hhX$02hhX$02hhX$02hhX$02hhX$02hhX$02hhX$02hhX$02hhX$02hhX$02hhX$02hhX$02hhX$02hhX$02hhX$02hhX$02hhX$02hhX$02hhX$02hhX$02hhX$02hhX$02hhX$02hhX$02hhX$02hhX$02hhX$02hhX$02hhX$02hhX$02hhX$02hhX$02hhX$02hhX$02hhX$02hhX$02hhX$02hhX$02hhX$02hhX$02hhX$02hhX$02hhX$02hhX$02hhX$02hhX$02hhX$02hhX$02hhX$02hhX$02hhX$02hhX$02hhX$02hhX$02hhX$02hhX$02hhX$02hhX$02hhX$02hhX$02hhX$02hhX$02hhX$02hhX$02hhX$02hhX$02hhX$02hhX$02hhX$02hhX$02hhX$02hhX$02hhX$02hhX$02hhX$02hhX$02hhX$02hhX$02hhX$02hhX$02hhX$02hhX$02hhX$02hhX$02hhX$02hhX$02hhX$02hhX$02hhX$02hhX$02hhX$02hhX$02hhX$02hhX$02hhX$02hhX$02hhX$02hhX$02hhX$02hhX$02hhX$02hhX$02hhX$02hh
68
                                                                                                      \& data1, \& data2, \& data3, \& data4a, data4b, data4b + 1, data4b + 2, data4b + 3, data4b + 4, \\
69
                                   data4b + 5) != 10)
70
                                                                return 0;
                                         memcpy(id->guid + 0x0, &data1, 4);
72
                                         memcpy(id->guid + 0x4, &data2, 2);
                                         memcpy(id->guid + 0x6, &data3, 2);
73
                                        memcpy(id->guid + 0x8, &data4a, 2);
memcpy(id->guid + 0xA, data4b, 6);
74
75
76
                                         return 1:
77 }
```

Here is the caller graph for this function:



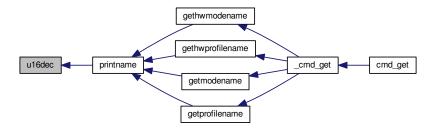
8.36.1.22 void u16dec ( ushort \* in, char \* out, size\_t \* srclen, size\_t \* dstlen )

Definition at line 105 of file profile.c.

References utf16to8.

Referenced by printname().

Here is the caller graph for this function:



8.36.1.23 void u16enc ( char \* in, ushort \* out, size\_t \* srclen, size\_t \* dstlen )

Definition at line 97 of file profile.c.

References utf8to16.

Referenced by cmd\_name(), and cmd\_profilename().

Here is the caller graph for this function:



8.36.1.24 void urldecode2 ( char \* dst, const char \* src )

Definition at line 8 of file profile.c.

Referenced by cmd\_name(), and cmd\_profilename().

```
8
        char a, b;
10
         char s;
          while ((s = \starsrc)) {
               if((s == '%') &&
	((a = src[1]) && (b = src[2])) &&
	(isxdigit(a) && isxdigit(b))){
12
13
14
                     if (a >= 'a')
a -= 'a'-'A';
15
                     if (a >= 'A')
a -= 'A' - 10;
17
18
                     else
19
                          a -= '0';
20
                     a -= '0';

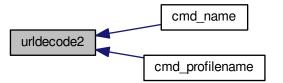
if(b >= 'a')

b -= 'a'-'A';

if(b >= 'A')

b -= 'A' - 10;
21
23
2.4
2.5
                     else
                          b -= '0';
26
                     *dst++ = 16 * a + b;
27
28
                     src += 3;
29
               } else {
                     *dst++ = s;
30
31
                     src++;
32
               }
33
34
          *dst = '\0';
35 }
```

Here is the caller graph for this function:

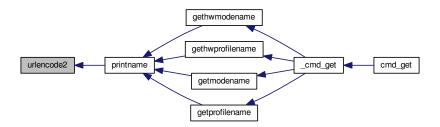


8.36.1.25 void urlencode2 ( char \* dst, const char \* src )

Definition at line 37 of file profile.c.

Referenced by printname().

```
37
38
        char s;
        while((s = *src++)){
           if(s <= ',' || s == '/' ||
(s >= ':' && s <= '@') ||
s == '[' || s == ']' ||
40
41
42
                       s >= 0x7F) {
43
                   char a = s \gg 4, b = s \& 0xF;
                   if(a >= 10)
                       a += 'A' - 10;
47
                   else
                   a += '0';
if (b >= 10)
48
49
                       b += 'A' - 10;
50
```



## 8.36.2 Variable Documentation

```
8.36.2.1 iconv_t utf16to8 = 0 [static]
```

Definition at line 95 of file profile.c.

Referenced by u16dec().

```
8.36.2.2 iconv_t utf8to16 = 0 [static]
```

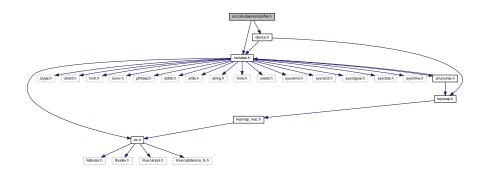
Definition at line 95 of file profile.c.

Referenced by u16enc().

# 8.37 src/ckb-daemon/profile.h File Reference

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

Include dependency graph for profile.h:



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



#### **Macros**

• #define hwloadprofile(kb, apply) (kb)->vtable->hwload(kb, 0, 0, apply, 0)

#### **Functions**

- void allocprofile (usbdevice \*kb)
- int loadprofile (usbdevice \*kb)
- void freeprofile (usbdevice \*kb)
- void cmd\_erase (usbdevice \*kb, usbmode \*mode, int dummy1, int dummy2, const char \*dummy3)
- void cmd\_eraseprofile (usbdevice \*kb, usbmode \*dummy1, int dummy2, int dummy3, const char \*dummy4)
- void cmd\_name (usbdevice \*kb, usbmode \*mode, int dummy1, int dummy2, const char \*name)
- void cmd\_profilename (usbdevice \*kb, usbmode \*dummy1, int dummy2, int dummy3, const char \*name)
- char \* getmodename (usbmode \*mode)
- char \* getprofilename (usbprofile \*profile)
- char \* gethwmodename (hwprofile \*profile, int index)
- char \* gethwprofilename (hwprofile \*profile)
- int setid (usbid \*id, const char \*guid)
- char \* getid (usbid \*id)
- void hwtonative (usbprofile \*profile, hwprofile \*hw, int modecount)
- void nativetohw (usbprofile \*profile, hwprofile \*hw, int modecount)
- void cmd\_id (usbdevice \*kb, usbmode \*mode, int dummy1, int dummy2, const char \*id)
- void cmd\_profileid (usbdevice \*kb, usbmode \*mode, int dummy1, int dummy2, const char \*id)
- int cmd\_hwload\_kb (usbdevice \*kb, usbmode \*dummy1, int dummy2, int apply, const char \*dummy3)
- int cmd\_hwload\_mouse (usbdevice \*kb, usbmode \*dummy1, int dummy2, int apply, const char \*dummy3)
- int cmd\_hwsave\_kb (usbdevice \*kb, usbmode \*dummy1, int dummy2, int dummy3, const char \*dummy4)
- int cmd\_hwsave\_mouse (usbdevice \*kb, usbmode \*dummy1, int dummy2, int dummy3, const char \*dummy4)

#### 8.37.1 Macro Definition Documentation

8.37.1.1 #define hwloadprofile( kb, apply ) (kb)->vtable->hwload(kb, 0, 0, apply, 0)

Definition at line 52 of file profile.h.

Referenced by \_start\_dev(), and loadprofile().

## 8.37.2 Function Documentation

8.37.2.1 void allocprofile ( usbdevice \* kb )

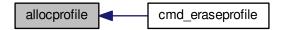
Definition at line 182 of file profile.c.

References usbprofile::currentmode, dpiset::forceupdate, lighting::forceupdate, initmode(), usbprofile::lastdpi, usbprofile::mode, MODE\_COUNT, and usbdevice::profile.

Referenced by cmd\_eraseprofile().



Here is the caller graph for this function:

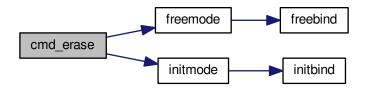


8.37.2.2 void cmd\_erase ( usbdevice \* kb, usbmode \* mode, int dummy1, int dummy2, const char \* dummy3 )

Definition at line 203 of file profile.c.

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

Here is the call graph for this function:



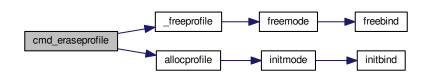
8.37.2.3 void cmd\_eraseprofile ( usbdevice \* kb, usbmode \* dummy1, int dummy2, int dummy3, const char \* dummy4 )

Definition at line 221 of file profile.c.

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

```
221
222    pthread_mutex_lock(imutex(kb));
223    _freeprofile(kb);
224    allocprofile(kb);
225    pthread_mutex_unlock(imutex(kb));
226 }
```

Here is the call graph for this function:



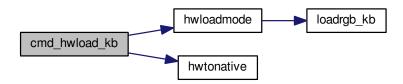
8.37.2.4 int cmd\_hwload\_kb ( usbdevice \* kb, usbmode \* dummy1, int dummy2, int apply, const char \* dummy3 )

Definition at line 16 of file profile\_keyboard.c.

References DELAY\_LONG, usbdevice::hw, hwloadmode(), HWMODE\_K70, HWMODE\_K95, hwtonative(), hwprofile::id, IS\_K95, MSG\_SIZE, hwprofile::name, PR\_NAME\_LEN, usbdevice::profile, and usbrecv.

```
16
17
       DELAY_LONG(kb);
       hwprofile* hw = calloc(1, sizeof(hwprofile));
18
      19
20
22
24
       uchar in_pkt[MSG_SIZE];
       int modes = (IS_K95(kb) ? HWMODE_K95 : HWMODE_K70);
for(int i = 0; i <= modes; i++) {</pre>
2.5
26
27
           data_pkt[0][3] = i;
           if(!usbrecv(kb, data_pkt[0], in_pkt)){
```

```
free(hw);
30
               return -1;
31
           memcpy(hw->id + i, in_pkt + 4, sizeof(usbid));
32
33
       // Ask for profile name
34
       if(!usbrecv(kb, data_pkt[1], in_pkt)){
35
36
           free(hw);
37
           return -1;
38
       memcpy(hw->name[0], in_pkt + 4, PR_NAME_LEN * 2);
39
       // Load modes
40
       for (int i = 0; i < modes; i++) {</pre>
41
42
           if (hwloadmode(kb, hw, i)) {
43
               free(hw);
44
               return -1;
45
           }
46
       // Make the profile active (if requested)
47
       if(apply)
49
           hwtonative(kb->profile, hw, modes);
50
       // Free the existing profile (if any)
51
       free (kb->hw);
       kb->hw = hw;
52
       DELAY_LONG(kb);
53
       return 0;
55 }
```



8.37.2.5 int cmd\_hwload\_mouse ( usbdevice \* kb, usbmode \* dummy1, int dummy2, int apply, const char \* dummy3 )

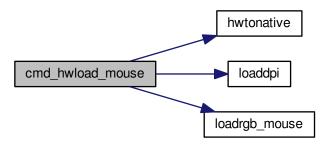
Definition at line 6 of file profile\_mouse.c.

References DELAY\_LONG, hwprofile::dpi, usbdevice::hw, hwtonative(), hwprofile::id, hwprofile::light, loaddpi(), loadrgb\_mouse(), MSG\_SIZE, hwprofile::name, PR\_NAME\_LEN, usbdevice::profile, and usbrecv.

```
6
                                                                                                                {
      DELAY_LONG(kb);
      hwprofile* hw = calloc(1, sizeof(hwprofile));
8
      // Ask for profile and mode IDs
       uchar data_pkt[2][MSG_SIZE] = {
10
            { 0x0e, 0x15, 0x01, 0 }, { 0x0e, 0x16, 0x01, 0 }
11
12
13
       uchar in_pkt[MSG_SIZE];
14
       for (int i = 0; i <= 1; i++) {
15
            data_pkt[0][3] = i;
16
            if(!usbrecv(kb, data_pkt[0], in_pkt)){
18
                free(hw);
19
20
            memcpy(hw->id + i, in_pkt + 4, sizeof(usbid));
21
23
        // Ask for profile and mode names
        for(int i = 0; i <= 1; i++) {
    data_pkt[1][3] = i;</pre>
25
2.6
            if(!usbrecv(kb, data_pkt[1],in_pkt)){
                 free (hw);
28
                 return -1;
            }
```

```
30
          memcpy(hw->name[i], in_pkt + 4, PR_NAME_LEN * 2);
32
      // Load the RGB and DPI settings
33
34
      35
36
          free(hw);
37
38
39
      \ensuremath{//} Make the profile active (if requested)
40
41
      if(apply)
          hwtonative(kb->profile, hw, 1);
42
43
      // Free the existing profile (if any)
44
      free(kb->hw);
      kb->hw = hw;
DELAY_LONG(kb);
45
46
      return 0;
47
48 }
```

Here is the call graph for this function:

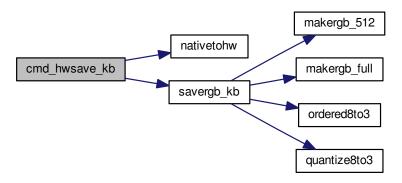


 $8.37.2.6 \quad \text{int cmd\_hwsave\_kb ( usbdevice} * \textit{kb, usbmode} * \textit{dummy1, int dummy2, int dummy3, const char} * \textit{dummy4} \text{)}$ 

Definition at line 57 of file profile\_keyboard.c.

References DELAY\_LONG, usbdevice::hw, HWMODE\_K70, HWMODE\_K95, hwprofile::id, IS\_K95, hwprofile::light, MD\_NAME\_LEN, MSG\_SIZE, hwprofile::name, nativetohw(), usbdevice::profile, savergb\_kb(), and usbsend.

```
57
        DELAY_LONG(kb);
59
        hwprofile* hw = kb->hw;
60
        if(!hw)
            hw = kb->hw = calloc(1, sizeof(hwprofile));
61
        int modes = (IS_K95(kb) ? HWMODE_K95 : HWMODE_K70);
62
        nativetohw(kb->profile, hw, modes);
        // Save the profile and mode names
6.5
        uchar data_pkt[2][MSG_SIZE] = {
             { 0x07, 0x16, 0x01, 0 },
{ 0x07, 0x15, 0x01, 0 },
66
67
68
        // Save the mode names
for(int i = 0; i <= modes; i++){</pre>
69
70
            data_pkt[0][3] = i;
memcpy(data_pkt[0] + 4, hw->name[i], MD_NAME_LEN * 2);
71
72
73
             if(!usbsend(kb, data_pkt[0], 1))
74
                 return -1;
75
76
        // Save the IDs
77
        for(int i = 0; i <= modes; i++) {</pre>
            data_pkt[1][3] = i;
memcpy(data_pkt[1] + 4, hw->id + i, sizeof(usbid));
78
79
80
             if(!usbsend(kb, data_pkt[1], 1))
                 return -1;
81
        }
```



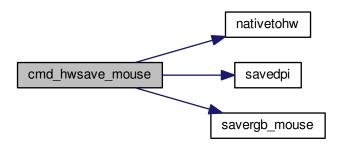
8.37.2.7 int cmd\_hwsave\_mouse ( usbdevice \* kb, usbmode \* dummy1, int dummy2, int dummy3, const char \* dummy4 )

Definition at line 50 of file profile\_mouse.c.

References DELAY\_LONG, hwprofile::dpi, usbdevice::hw, hwprofile::id, hwprofile::light, MD\_NAME\_LEN, MSG\_S-IZE, hwprofile::name, nativetohw(), usbdevice::profile, savedpi(), savergb\_mouse(), and usbsend.

```
50
       DELAY_LONG(kb);
52
       hwprofile* hw = kb->hw;
53
       if(!hw)
       hw = kb->hw = calloc(1, sizeof(hwprofile));
nativetohw(kb->profile, hw, 1);
54
55
56
       // Save the profile and mode names
       uchar data_pkt[2][MSG_SIZE] = {
            { 0x07, 0x16, 0x01, 0 },
{ 0x07, 0x15, 0x01, 0 },
59
60
       for (int i = 0; i \le 1; i++) {
61
           data_pkt[0][3] = i;
62
           memcpy(data_pkt[0] + 4, hw->name[i], MD_NAME_LEN * 2);
63
64
            if(!usbsend(kb, data_pkt[0], 1))
6.5
                return -1;
66
       // Save the IDs
67
       for(int i = 0; i <= 1; i++) {</pre>
68
           data_pkt[1][3] = i;
69
70
            memcpy(data_pkt[1] + 4, hw->id + i, sizeof(usbid));
71
            if(!usbsend(kb, data_pkt[1], 1))
72
                return -1:
73
       ^{\prime} // Save the RGB data for the non-DPI zones
74
75
       if(savergb_mouse(kb, hw->light, 0))
76
            return -1;
       // Save the DPI data (also saves RGB for those states)
78
       if(savedpi(kb, hw->dpi, hw->light))
79
            return -1:
80
       DELAY_LONG(kb);
81
       return 0;
82 }
```

Here is the call graph for this function:



8.37.2.8 void cmd\_id ( usbdevice \* kb, usbmode \* mode, int dummy1, int dummy2, const char \* id )

Definition at line 160 of file profile.c.

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

```
160

// ID is either a GUID or an 8-digit hex number

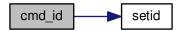
162 int newmodified;

163 if(!setid(&mode->id, id) && sscanf(id, "%08x", &newmodified) == 1)

164 memcpy(mode->id.modified, &newmodified, sizeof(newmodified));

165 }
```

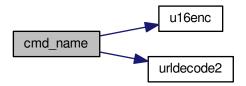
Here is the call graph for this function:



8.37.2.9 void cmd\_name ( usbdevice \* kb, usbmode \* mode, int dummy1, int dummy2, const char \* name )

Definition at line 117 of file profile.c.

References MD\_NAME\_LEN, usbmode::name, u16enc(), and urldecode2().



8.37.2.10 void cmd\_profileid ( usbdevice \* kb, usbmode \* mode, int dummy1, int dummy2, const char \* id )

Definition at line 167 of file profile.c.

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

```
167
168 usbprofile* profile = kb->profile;
169 int newmodified;
170 if(!setid(&profile->id, id) && sscanf(id, "%08x", &newmodified) == 1)
171 memcpy(profile->id.modified, &newmodified, sizeof(newmodified));
172
173 }
```

Here is the call graph for this function:



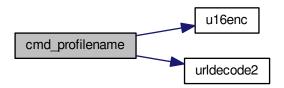
8.37.2.11 void cmd\_profilename ( usbdevice \* kb, usbmode \* dummy1, int dummy2, int dummy3, const char \* name )

Definition at line 124 of file profile.c.

References usbprofile::name, PR\_NAME\_LEN, usbdevice::profile, u16enc(), and urldecode2().

```
124
125    usbprofile* profile = kb->profile;
126    char decoded[strlen(name) + 1];
127    urldecode2(decoded, name);
128    size_t srclen = strlen(decoded), dstlen = PR_NAME_LEN;
129    ul6enc(decoded, profile->name, &srclen, &dstlen);
130 }
```

Here is the call graph for this function:



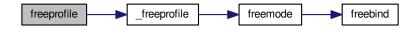
8.37.2.12 void freeprofile ( usbdevice \*kb )

Definition at line 228 of file profile.c.

References \_freeprofile(), and usbdevice::hw.

```
228
229    _freeprofile(kb);
230    // Also free HW profile
231    free(kb->hw);
232    kb->hw = 0;
233 }
```

Here is the call graph for this function:



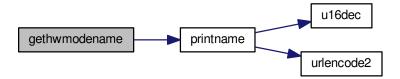
8.37.2.13 char\* gethwmodename ( hwprofile \* profile, int index )

Definition at line 152 of file profile.c.

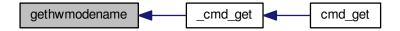
References MD\_NAME\_LEN, hwprofile::name, and printname().

Referenced by \_cmd\_get().

```
152 {
153     return printname(profile->name[index + 1], MD_NAME_LEN);
154 }
```



Here is the caller graph for this function:



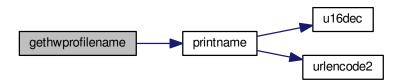
## 8.37.2.14 char\* gethwprofilename ( hwprofile \* profile )

Definition at line 156 of file profile.c.

References MD\_NAME\_LEN, hwprofile::name, and printname().

Referenced by \_cmd\_get().

Here is the call graph for this function:



Here is the caller graph for this function:



```
8.37.2.15 char* getid ( usbid * id )
```

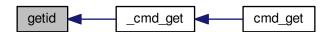
Definition at line 79 of file profile.c.

References usbid::guid.

Referenced by \_cmd\_get().

```
79
80
        int32_t data1;
        int16_t data2, data3, data4a;
        char data4b[6];
83
        memcpy(&data1, id->guid + 0x0, 4);
        memcpy(&data2, id->guid + 0x4, 2);
memcpy(&data3, id->guid + 0x6, 2);
84
85
        memcpy(&data4a, id->guid + 0x8, 2);
memcpy(&data4b, id->guid + 0xA, 6);
86
        char* guid = malloc(39);
snprintf(guid, 39, "{*08X-*04hX-*04hX-*02hhX*02hhX*02hhX*02hhX*02hhX*02hhX}",
89
                   data1, data2, data3, data4a, data4b[0], data4b[1], data4b[2], data4b[3], data4b[4], data4b[5])
90
91
        return guid;
92 }
```

Here is the caller graph for this function:



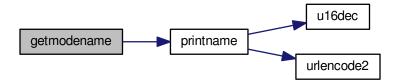
```
8.37.2.16 char* getmodename ( usbmode * mode )
```

Definition at line 144 of file profile.c.

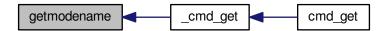
References MD NAME LEN, usbmode::name, and printname().

Referenced by \_cmd\_get().

```
144
145 return printname(mode->name, MD_NAME_LEN);
146 }
```



Here is the caller graph for this function:



```
8.37.2.17 char* getprofilename ( usbprofile * profile )
```

Definition at line 148 of file profile.c.

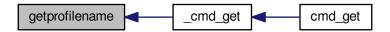
References usbprofile::name, PR\_NAME\_LEN, and printname().

Referenced by \_cmd\_get().

Here is the call graph for this function:



Here is the caller graph for this function:



8.37.2.18 void hwtonative ( usbprofile \* profile, hwprofile \* hw, int modecount )

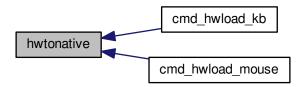
Definition at line 235 of file profile.c.

References usbmode::dpi, hwprofile::dpi, dpiset::forceupdate, lighting::forceupdate, usbmode::id, usbprofile::id, hwprofile::lastdpi, usbprofile::lastdpi, usbmode::light, hwprofile::light, MD\_NAME\_LEN, usbprofile::mode, usbmode::name, usbprofile::name, hwprofile::name, and PR NAME LEN.

Referenced by cmd\_hwload\_kb(), and cmd\_hwload\_mouse().

```
235
236
        // Copy the profile name and ID
237
        memcpy(profile->name, hw->name[0], PR_NAME_LEN * 2);
238
        memcpy(&profile->id, hw->id, sizeof(usbid));
239
        // Copy the mode settings
        for(int i = 0; i < modecount; i++){
    usbmode* mode = profile->mode + i;
240
241
            memcpy(mode->name, hw->name[i + 1], MD_NAME_LEN * 2);
242
243
            memcpy(&mode->id, hw->id + i + 1, sizeof(usbid));
244
            memcpy(&mode->light, hw->light + i, sizeof(lighting));
245
             memcpy(\&mode->dpi, hw->dpi + i, sizeof(dpiset));
             // Set a force update on the light/DPI since they've been overwritten
246
247
            mode->light.forceupdate = mode->dpi.forceupdate = 1;
248
249
        profile->lastlight.forceupdate = profile->lastdpi.
      forceupdate = 1;
250 }
```

Here is the caller graph for this function:



8.37.2.19 int loadprofile ( usbdevice \* kb )

Definition at line 192 of file profile.c.

References hwloadprofile.

```
192
193
if (hwloadprofile(kb, 1))
194
return -1;
195
return 0;
196}
```

8.37.2.20 void nativetohw ( usbprofile \* profile, hwprofile \* hw, int modecount )

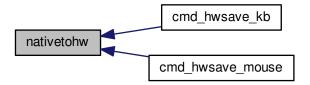
Definition at line 252 of file profile.c.

References usbmode::dpi, hwprofile::dpi, usbmode::id, usbprofile::id, hwprofile::id, usbmode::light, hwprofile::light, MD NAME LEN, usbprofile::mame, usbprofile::name, hwprofile::name, and PR NAME LEN.

Referenced by cmd\_hwsave\_kb(), and cmd\_hwsave\_mouse().

```
252
253
          // Copy name and {\tt ID}
          memcpy(hw->name[0], profile->name, PR_NAME_LEN * 2);
254
          memcpy(hw->id, &profile->id, sizeof(usbid));
255
256
          // Copy the mode settings
          for(int i = 0; i < modecount; i++) {</pre>
258
               usbmode* mode = profile->mode + i;
259
               memcpy(hw->name[i + 1], mode->name, MD_NAME_LEN * 2);
               memcpy(hw->id + i + 1, &mode->id, sizeof(usbid));
memcpy(hw->light + i, &mode->light, sizeof(lighting));
memcpy(hw->dpi + i, &mode->dpi, sizeof(dpiset));
260
2.61
262
263
          }
264 }
```

Here is the caller graph for this function:



```
8.37.2.21 int setid ( usbid *id, const char *guid )
```

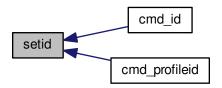
Definition at line 64 of file profile.c.

References usbid::guid.

Referenced by cmd\_id(), and cmd\_profileid().

```
64
65
                                       int32_t data1;
                                        int16_t data2, data3, data4a;
                                       char data4b[6];
if(sscanf(guid, "{%08X-%04hX-%04hX-%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02hhX%02h
67
68
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);
                                      memcpy(id->guid + 0x8, &data4a, 2);
memcpy(id->guid + 0xA, data4b, 6);
74
75
76
                                       return 1;
```

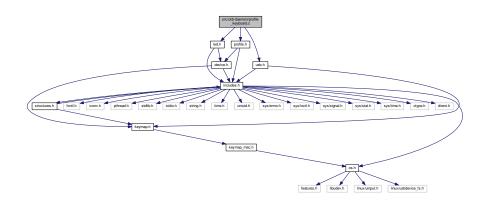
Here is the caller graph for this function:



# 8.38 src/ckb-daemon/profile\_keyboard.c File Reference

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

Include dependency graph for profile\_keyboard.c:



#### **Functions**

- static int hwloadmode (usbdevice \*kb, hwprofile \*hw, int mode)
- int cmd\_hwload\_kb (usbdevice \*kb, usbmode \*dummy1, int dummy2, int apply, const char \*dummy3)
- int cmd\_hwsave\_kb (usbdevice \*kb, usbmode \*dummy1, int dummy2, int dummy3, const char \*dummy4)

## 8.38.1 Function Documentation

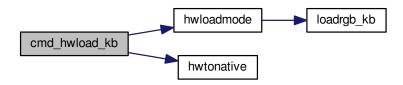
8.38.1.1 int cmd\_hwload\_kb ( usbdevice \* kb, usbmode \* dummy1, int dummy2, int apply, const char \* dummy3 )

Definition at line 16 of file profile\_keyboard.c.

References DELAY\_LONG, usbdevice::hw, hwloadmode(), HWMODE\_K70, HWMODE\_K95, hwtonative(), hwprofile::id, IS K95, MSG SIZE, hwprofile::name, PR NAME LEN, usbdevice::profile, and usbrecv.

```
16
17    DELAY_LONG(kb);
18    hwprofile* hw = calloc(1, sizeof(hwprofile));
19    // Ask for profile and mode IDs
```

```
20
        uchar data_pkt[2][MSG_SIZE] = {
            { 0x0e, 0x15, 0x01, 0 }, { 0x0e, 0x16, 0x01, 0 }
22
2.3
       uchar in_pkt[MSG_SIZE];
2.4
        int modes = (IS_K95(kb) ? HWMODE_K95 : HWMODE_K70);
for(int i = 0; i <= modes; i++) {</pre>
25
26
27
            data_pkt[0][3] = i;
28
            if(!usbrecv(kb, data_pkt[0], in_pkt)){
29
                 free(hw);
30
                 return -1;
31
            memcpy(hw->id + i, in_pkt + 4, sizeof(usbid));
32
33
34
        // Ask for profile name
35
        if(!usbrecv(kb, data_pkt[1], in_pkt)){
36
            free (hw);
37
            return -1;
38
39
        memcpy(hw->name[0], in_pkt + 4, PR_NAME_LEN * 2);
40
        // Load modes
41
        for(int i = 0; i < modes; i++) {</pre>
            if (hwloadmode(kb, hw, i)) {
42
4.3
                 free (hw);
44
                 return -1;
45
            }
46
47
        // Make the profile active (if requested)
48
        if(apply)
49
            hwtonative(kb->profile, hw, modes);
        // Free the existing profile (if any)
50
        free(kb->hw);
51
        kb->hw = hw;
53
        DELAY_LONG(kb);
        return 0;
55 }
```



8.38.1.2 int cmd\_hwsave\_kb ( usbdevice \* kb, usbmode \* dummy1, int dummy2, int dummy3, const char \* dummy4 )

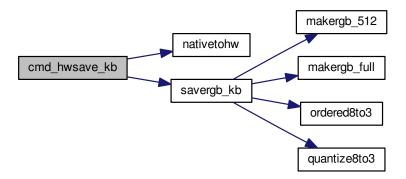
Definition at line 57 of file profile\_keyboard.c.

References DELAY\_LONG, usbdevice::hw, HWMODE\_K70, HWMODE\_K95, hwprofile::id, IS\_K95, hwprofile::light, MD\_NAME\_LEN, MSG\_SIZE, hwprofile::name, nativetohw(), usbdevice::profile, savergb\_kb(), and usbsend.

```
57
58
        DELAY_LONG(kb);
        hwprofile* hw = kb->hw;
60
        hw = kb->hw = calloc(1, sizeof(hwprofile));
int modes = (IS_K95(kb) ? HWMODE_K95 : HWMODE_K70);
61
62
        nativetohw(kb->profile, hw, modes);
63
        // Save the profile and mode names
        uchar data_pkt[2][MSG_SIZE] = {
             { 0x07, 0x16, 0x01, 0 },
             { 0x07, 0x15, 0x01, 0 },
67
68
        };
// Save the mode names
69
70
        for(int i = 0; i <= modes; i++) {</pre>
             data_pkt[0][3] = i;
```

```
memcpy(data_pkt[0] + 4, hw->name[i], MD_NAME_LEN * 2);
          if(!usbsend(kb, data_pkt[0], 1))
74
              return -1;
75
      // Save the IDs
for(int i = 0; i <= modes; i++){
76
          data_pkt[1][3] = i;
78
79
          memcpy(data_pkt[1] + 4, hw->id + i, sizeof(usbid));
80
          if(!usbsend(kb, data_pkt[1], 1))
81
              return -1;
82
      83
84
85
86
              return -1;
87
      DELAY_LONG(kb);
88
89
      return 0;
90 }
```

Here is the call graph for this function:



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

Definition at line 5 of file profile\_keyboard.c.

References hwprofile::light, loadrgb\_kb(), MD\_NAME\_LEN, MSG\_SIZE, hwprofile::name, and usbrecv.

Referenced by cmd\_hwload\_kb().



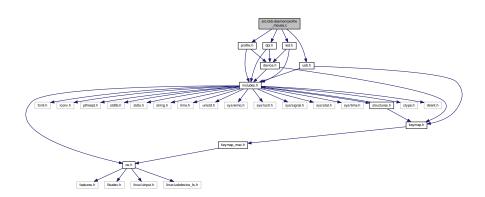
Here is the caller graph for this function:



# 8.39 src/ckb-daemon/profile\_mouse.c File Reference

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

Include dependency graph for profile\_mouse.c:



## **Functions**

- int cmd\_hwload\_mouse (usbdevice \*kb, usbmode \*dummy1, int dummy2, int apply, const char \*dummy3)
- int cmd\_hwsave\_mouse (usbdevice \*kb, usbmode \*dummy1, int dummy2, int dummy3, const char \*dummy4)

## 8.39.1 Function Documentation

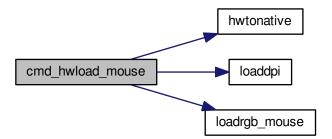
8.39.1.1 int cmd\_hwload\_mouse ( usbdevice \* kb, usbmode \* dummy1, int dummy2, int apply, const char \* dummy3 )

Definition at line 6 of file profile mouse.c.

References DELAY\_LONG, hwprofile::dpi, usbdevice::hw, hwtonative(), hwprofile::id, hwprofile::light, loaddpi(), loadrgb\_mouse(), MSG\_SIZE, hwprofile::name, PR\_NAME\_LEN, usbdevice::profile, and usbrecv.

```
DELAY_LONG(kb);
hwprofile* hw = calloc(1, sizeof(hwprofile));
7
8
      10
11
12
13
       uchar in_pkt[MSG_SIZE];
14
       for(int i = 0; i <= 1; i++) {
    data_pkt[0][3] = i;</pre>
15
16
           if(!usbrecv(kb, data_pkt[0], in_pkt)){
18
               free(hw);
19
20
           memcpy(hw->id + i, in_pkt + 4, sizeof(usbid));
21
22
       // Ask for profile and mode names
23
       for(int i = 0; i <= 1; i++) {
    data_pkt[1][3] = i;</pre>
25
26
           if(!usbrecv(kb, data_pkt[1],in_pkt)){
2.7
                free (hw);
28
               return -1:
29
           memcpy(hw->name[i], in_pkt + 4, PR_NAME_LEN * 2);
31
32
33
       \ensuremath{//} Load the RGB and DPI settings
       34
35
36
37
           return -1;
38
39
       // Make the profile active (if requested)
40
41
       if(apply)
           hwtonative(kb->profile, hw, 1);
       // Free the existing profile (if any)
44
       free(kb->hw);
       kb->hw = hw;
DELAY_LONG(kb);
45
46
       return 0;
48 }
```

Here is the call graph for this function:



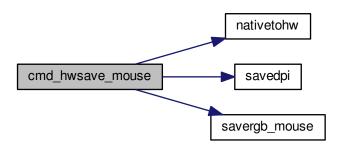
8.39.1.2 int cmd\_hwsave\_mouse ( usbdevice \* kb, usbmode \* dummy1, int dummy2, int dummy3, const char \* dummy4 )

Definition at line 50 of file profile\_mouse.c.

References DELAY\_LONG, hwprofile::dpi, usbdevice::hw, hwprofile::id, hwprofile::light, MD\_NAME\_LEN, MSG\_S-IZE, hwprofile::name, nativetohw(), usbdevice::profile, savedpi(), savergb\_mouse(), and usbsend.

```
50
         DELAY_LONG(kb);
51
         hwprofile* hw = kb->hw;
52
53
         if(!hw)
        hw = kb->hw = calloc(1, sizeof(hwprofile));
nativetohw(kb->profile, hw, 1);
55
56
         // Save the profile and mode names
57
        uchar data_pkt[2][MSG_SIZE] = {
              { 0x07, 0x16, 0x01, 0 },
{ 0x07, 0x15, 0x01, 0 },
58
59
60
         for(int i = 0; i <= 1; i++){</pre>
              data_pkt[0][3] = i;
memcpy(data_pkt[0] + 4, hw->name[i], MD_NAME_LEN * 2);
63
              if(!usbsend(kb, data_pkt[0], 1))
64
65
                   return -1;
66
         // Save the IDs
        for(int i = 0; i <= 1; i++) {
    data_pkt[1][3] = i;
    memcpy(data_pkt[1] + 4, hw->id + i, sizeof(usbid));
    if(!usbsend(kb, data_pkt[1], 1))
68
69
70
71
72
                   return -1;
        ^{\prime} // Save the RGB data for the non-DPI zones
74
75
        if(savergb_mouse(kb, hw->light, 0))
76
              return -1;
         // Save the DPI data (also saves RGB for those states)
        if(savedpi(kb, hw->dpi, hw->light))
78
80
         DELAY_LONG(kb);
81
         return 0;
82 }
```

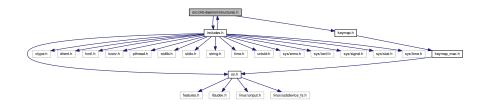
Here is the call graph for this function:



## 8.40 src/ckb-daemon/structures.h File Reference

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

Include dependency graph for structures.h:



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



## **Data Structures**

- struct usbid
- · struct macroaction
- · struct keymacro
- · struct binding
- · struct dpiset
- · struct lighting
- struct usbmode
- struct usbprofile
- struct hwprofile
- struct usbinputstruct usbdevice

## Macros

- #define SET\_KEYBIT(array, index) do { (array)[(index) / 8] |= 1 << ((index) % 8); } while(0)</li>
- #define CLEAR\_KEYBIT(array, index) do { (array)[(index) / 8] &= ~(1 << ((index) % 8)); } while(0)</li>
- #define I\_NUM 1
- #define I\_CAPS 2
- #define I SCROLL 4
- #define OUTFIFO\_MAX 10
- #define MACRO\_MAX 1024
- #define DPI\_COUNT 6
- #define LIFT\_MIN 1
- #define LIFT\_MAX 5
- #define MD\_NAME\_LEN 16
- #define PR\_NAME\_LEN 16
- #define MODE\_COUNT 6
- #define HWMODE\_K70 1
- #define HWMODE\_K95 3
- #define HWMODE\_MAX 3
- #define FEAT\_RGB 0x001
- #define FEAT\_MONOCHROME 0x002
- #define FEAT\_POLLRATE 0x004

- #define FEAT\_ADJRATE 0x008
- #define FEAT\_BIND 0x010
- #define FEAT NOTIFY 0x020
- #define FEAT\_FWVERSION 0x040
- #define FEAT\_FWUPDATE 0x080
- #define FEAT\_HWLOAD 0x100
- #define FEAT\_ANSI 0x200
- #define FEAT\_ISO 0x400
- #define FEAT MOUSEACCEL 0x800
- #define FEAT\_COMMON (FEAT\_BIND | FEAT\_NOTIFY | FEAT\_FWVERSION | FEAT\_MOUSEACCEL | FEAT\_HWLOAD)
- #define FEAT\_STD\_RGB (FEAT\_COMMON | FEAT\_RGB | FEAT\_POLLRATE | FEAT\_FWUPDATE)
- #define FEAT\_STD\_NRGB (FEAT\_COMMON)
- #define FEAT\_LMASK (FEAT\_ANSI | FEAT\_ISO)
- #define HAS FEATURES(kb, feat) (((kb)->features & (feat)) == (feat))
- #define HAS ANY FEATURE(kb, feat) (!!((kb)->features & (feat)))
- #define NEEDS\_FW\_UPDATE(kb) ((kb)->fwversion == 0 && HAS\_FEATURES((kb), FEAT\_FWUPDATE |
  FEAT\_FWVERSION))
- #define SCROLL\_ACCELERATED 0
- #define SCROLL MIN 1
- #define SCROLL MAX 10
- #define KB\_NAME\_LEN 40
- #define SERIAL\_LEN 34
- #define MSG\_SIZE 64
- #define IFACE\_MAX 4

#### Variables

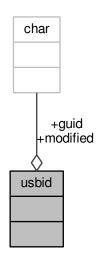
- · const union devcmd vtable\_keyboard
- const union devcmd vtable keyboard nonrgb
- · const union devcmd vtable\_mouse

#### 8.40.1 Data Structure Documentation

#### 8.40.1.1 struct usbid

Definition at line 8 of file structures.h.

Collaboration diagram for usbid:



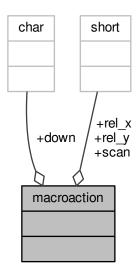
## **Data Fields**

char	guid[16]	
char	modified[4]	

8.40.1.2 struct macroaction

Definition at line 27 of file structures.h.

Collaboration diagram for macroaction:



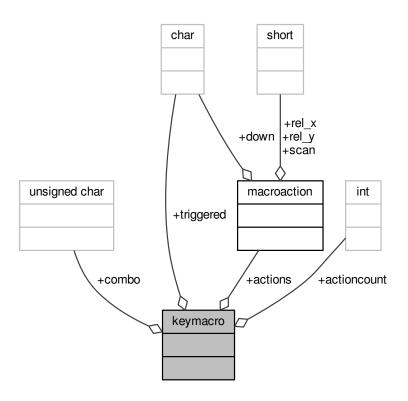
## Data Fields

char	down	
short	rel_x	
short	rel_y	
short	scan	

8.40.1.3 struct keymacro

Definition at line 34 of file structures.h.

Collaboration diagram for keymacro:



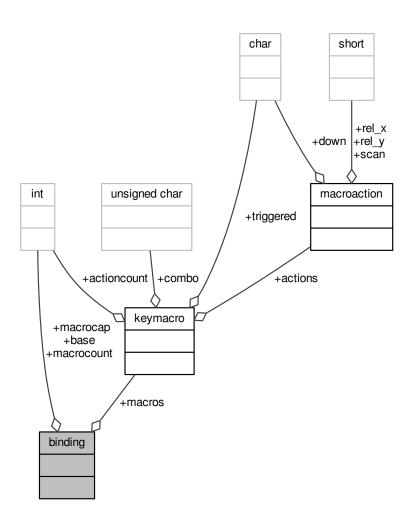
## Data Fields

int	actioncount	
macroaction *	actions	
uchar	combo[((((152+3+	2)+25)+7)/8)]
char	triggered	

## 8.40.1.4 struct binding

Definition at line 42 of file structures.h.

Collaboration diagram for binding:



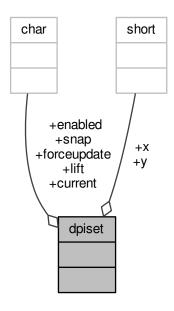
# Data Fields

int	base[((152+3+12)+2	25)]
int	macrocap	
int	macrocount	
keymacro *	macros	

## 8.40.1.5 struct dpiset

Definition at line 56 of file structures.h.

Collaboration diagram for dpiset:



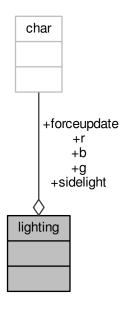
# Data Fields

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

8.40.1.6 struct lighting

Definition at line 72 of file structures.h.

Collaboration diagram for lighting:



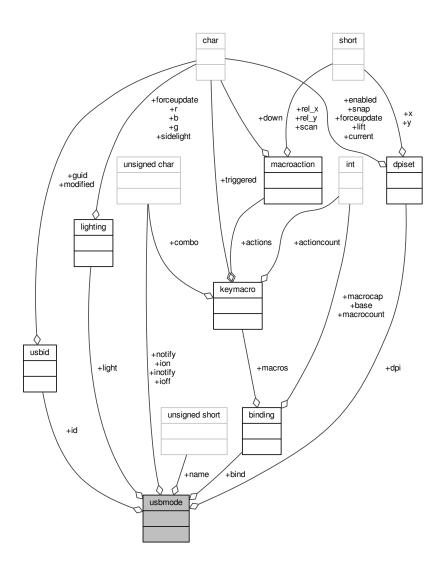
## Data Fields

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

8.40.1.7 struct usbmode

Definition at line 82 of file structures.h.

## Collaboration diagram for usbmode:



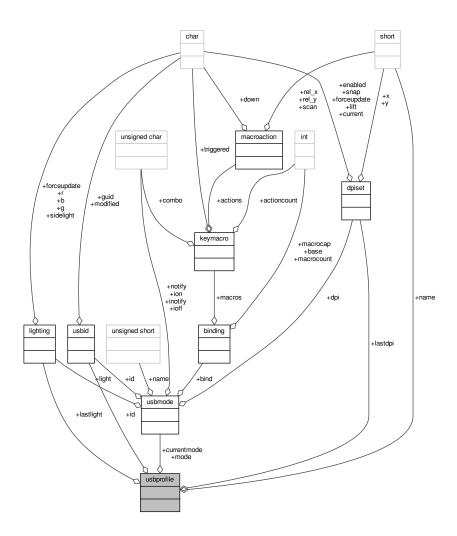
#### **Data Fields**

binding	bind
dpiset	dpi
usbid	id
uchar	inotify[10]
uchar	ioff
uchar	ion
lighting	light
ushort	name[16]
uchar	notify[10][((((152+3+12)+25)+7)/8)]

## 8.40.1.8 struct usbprofile

Definition at line 100 of file structures.h.

Collaboration diagram for usbprofile:



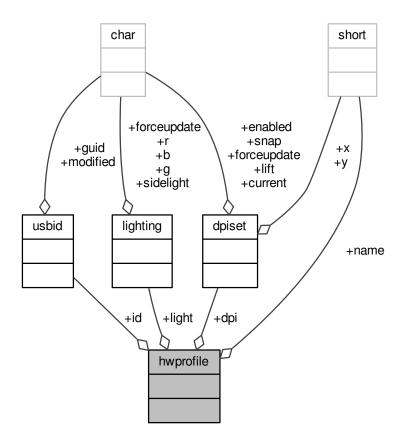
### Data Fields

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

# 8.40.1.9 struct hwprofile

Definition at line 117 of file structures.h.

Collaboration diagram for hwprofile:



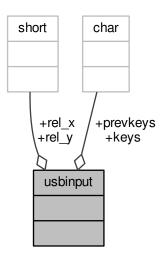
# **Data Fields**

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

8.40.1.10 struct usbinput

Definition at line 128 of file structures.h.

Collaboration diagram for usbinput:



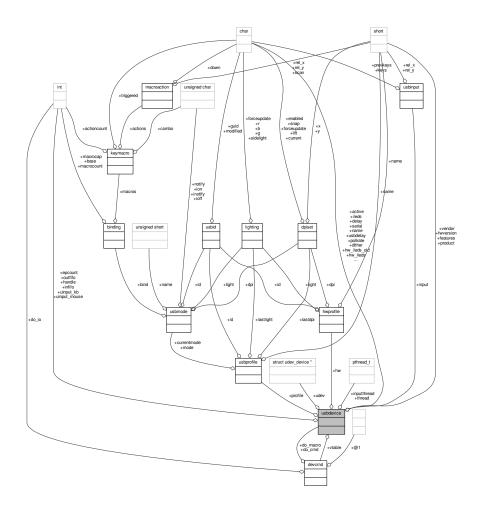
### **Data Fields**

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

8.40.1.11 struct usbdevice

Definition at line 177 of file structures.h.

# Collaboration diagram for usbdevice:



# Data Fields

char	active	
char	delay	
char	dither	
int	epcount	
ushort	features	
ushort	fwversion	
int	handle	
hwprofile *	hw	
uchar	hw_ileds	
uchar	hw_ileds_old	
uchar	ileds	
int	infifo	
usbinput	input	
pthread_t	inputthread	
char	name[40+1]	

int	outfifo[10]	
char	pollrate	
short	product	
usbprofile *	profile	
char	serial[34]	
pthread_t	thread	
struct	udev	
udev_device *		
int	uinput_kb	
int	uinput_mouse	
char	usbdelay	
short	vendor	
const union	vtable	
devcmd *		

### 8.40.2 Macro Definition Documentation

8.40.2.1 #define CLEAR\_KEYBIT( array, index) do { (array)[(index) / 8] &=  $\sim$ (1 << ((index) % 8)); } while(0)

Definition at line 16 of file structures.h.

Referenced by cmd\_notify(), corsair\_mousecopy(), hid\_kb\_translate(), and hid\_mouse\_translate().

8.40.2.2 #define DPI\_COUNT 6

Definition at line 53 of file structures.h.

Referenced by cmd\_dpi(), cmd\_dpisel(), loaddpi(), printdpi(), savedpi(), and updatedpi().

8.40.2.3 #define FEAT\_ADJRATE 0x008

Definition at line 138 of file structures.h.

Referenced by \_mkdevpath(), \_setupusb(), and \_start\_dev().

8.40.2.4 #define FEAT\_ANSI 0x200

Definition at line 145 of file structures.h.

Referenced by readcmd().

8.40.2.5 #define FEAT\_BIND 0x010

Definition at line 139 of file structures.h.

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

8.40.2.6 #define FEAT\_COMMON (FEAT\_BIND | FEAT\_NOTIFY | FEAT\_FWVERSION | FEAT\_MOUSEACCEL | FEAT\_HWLOAD)

Definition at line 150 of file structures.h.

8.40.2.7 #define FEAT\_FWUPDATE 0x080

Definition at line 142 of file structures.h.

Referenced by \_mkdevpath(), \_start\_dev(), and cmd\_fwupdate().

8.40.2.8 #define FEAT\_FWVERSION 0x040

Definition at line 141 of file structures.h.

Referenced by \_mkdevpath(), and \_start\_dev().

8.40.2.9 #define FEAT\_HWLOAD 0x100

Definition at line 143 of file structures.h.

Referenced by \_start\_dev().

8.40.2.10 #define FEAT\_ISO 0x400

Definition at line 146 of file structures.h.

Referenced by readcmd().

8.40.2.11 #define FEAT\_LMASK (FEAT\_ANSI | FEAT\_ISO)

Definition at line 153 of file structures.h.

Referenced by readcmd().

8.40.2.12 #define FEAT\_MONOCHROME 0x002

Definition at line 136 of file structures.h.

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

8.40.2.13 #define FEAT\_MOUSEACCEL 0x800

Definition at line 147 of file structures.h.

Referenced by main(), and readcmd().

8.40.2.14 #define FEAT\_NOTIFY 0x020

Definition at line 140 of file structures.h.

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

8.40.2.15 #define FEAT\_POLLRATE 0x004

Definition at line 137 of file structures.h.

Referenced by \_mkdevpath(), \_start\_dev(), and getfwversion().

8.40.2.16 #define FEAT\_RGB 0x001

Definition at line 135 of file structures.h.

Referenced by \_mkdevpath(), \_start\_dev(), os\_setupusb(), revertusb(), and usbunclaim().

8.40.2.17 #define FEAT\_STD\_NRGB (FEAT\_COMMON)

Definition at line 152 of file structures.h.

Referenced by \_setupusb().

8.40.2.18 #define FEAT\_STD\_RGB (FEAT\_COMMON | FEAT\_RGB | FEAT\_POLLRATE | FEAT\_FWUPDATE)

Definition at line 151 of file structures.h.

Referenced by \_setupusb().

8.40.2.19 #define HAS\_ANY\_FEATURE( kb, feat ) (!!((kb)->features & (feat)))

Definition at line 157 of file structures.h.

8.40.2.20 #define HAS\_FEATURES( kb, feat ) (((kb)->features & (feat)) == (feat))

Definition at line 156 of file structures.h.

Referenced by \_mkdevpath(), \_start\_dev(), cmd\_fwupdate(), os\_setupusb(), readcmd(), revertusb(), and usbunclaim().

8.40.2.21 #define HWMODE\_K70 1

Definition at line 114 of file structures.h.

Referenced by cmd\_hwload\_kb(), and cmd\_hwsave\_kb().

8.40.2.22 #define HWMODE\_K95 3

Definition at line 115 of file structures.h.

Referenced by cmd\_hwload\_kb(), and cmd\_hwsave\_kb().

8.40.2.23 #define HWMODE\_MAX 3

Definition at line 116 of file structures.h.

8.40.2.24 #define I\_CAPS 2

Definition at line 20 of file structures.h.

Referenced by \_cmd\_get(), iselect(), nprintind(), and updateindicators\_kb().

8.40.2.25 #define I\_NUM 1

Definition at line 19 of file structures.h.

Referenced by \_cmd\_get(), iselect(), nprintind(), and updateindicators\_kb().

8.40.2.26 #define I\_SCROLL 4

Definition at line 21 of file structures.h.

Referenced by \_cmd\_get(), iselect(), nprintind(), and updateindicators\_kb().

8.40.2.27 #define IFACE\_MAX 4

Definition at line 176 of file structures.h.

8.40.2.28 #define KB\_NAME\_LEN 40

Definition at line 173 of file structures.h.

Referenced by \_setupusb(), and os\_setupusb().

8.40.2.29 #define LIFT\_MAX 5

Definition at line 55 of file structures.h.

Referenced by cmd\_lift(), and loaddpi().

8.40.2.30 #define LIFT\_MIN 1

Definition at line 54 of file structures.h.

Referenced by cmd\_lift(), and loaddpi().

8.40.2.31 #define MACRO\_MAX 1024

Definition at line 50 of file structures.h.

Referenced by \_cmd\_macro().

8.40.2.32 #define MD\_NAME\_LEN 16

Definition at line 81 of file structures.h.

Referenced by cmd\_hwsave\_kb(), cmd\_hwsave\_mouse(), cmd\_name(), gethwmodename(), gethwprofilename(), gethwdename(), hwloadmode(), hwtonative(), and nativetohw().

8.40.2.33 #define MODE\_COUNT 6

Definition at line 99 of file structures.h.

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

8.40.2.34 #define MSG\_SIZE 64

Definition at line 175 of file structures.h.

Referenced by \_usbsend(), cmd\_hwload\_kb(), cmd\_hwload\_mouse(), cmd\_hwsave\_kb(), cmd\_hwsave\_mouse(), cmd\_pollrate(), fwupdate(), getfwversion(), hwloadmode(), loaddpi(), loadrgb\_kb(), loadrgb\_mouse(), os\_inputmain(), os\_usbrecv(), os\_usbsend(), savedpi(), savergb\_kb(), savergb\_mouse(), setactive\_kb(), setactive\_mouse(), updatedpi(), updatergb\_kb(), and updatergb\_mouse().

8.40.2.35 #define NEEDS\_FW\_UPDATE( kb ) ((kb)->fwversion == 0 && HAS\_FEATURES((kb), FEAT\_FWUPDATE | FEAT\_FWVERSION))

Definition at line 160 of file structures.h.

Referenced by \_start\_dev(), readcmd(), revertusb(), setactive\_kb(), and setactive\_mouse().

8.40.2.36 #define OUTFIFO\_MAX 10

Definition at line 24 of file structures.h.

Referenced by \_mknotifynode(), \_rmnotifynode(), inputupdate\_keys(), nprintf(), readcmd(), rmdevpath(), and updateindicators\_kb().

8.40.2.37 #define PR\_NAME\_LEN 16

Definition at line 98 of file structures.h.

Referenced by cmd\_hwload\_kb(), cmd\_hwload\_mouse(), cmd\_profilename(), getprofilename(), hwtonative(), and nativetohw().

8.40.2.38 #define SCROLL\_ACCELERATED 0

Definition at line 163 of file structures.h.

Referenced by readcmd().

8.40.2.39 #define SCROLL MAX 10

Definition at line 165 of file structures.h.

Referenced by readcmd().

8.40.2.40 #define SCROLL\_MIN 1

Definition at line 164 of file structures.h.

Referenced by readcmd().

8.40.2.41 #define SERIAL\_LEN 34

Definition at line 174 of file structures.h.

Referenced by \_setupusb(), and os\_setupusb().

8.40.2.42 #define SET\_KEYBIT( array, index ) do { (array)[(index) / 8] |= 1 << ((index) % 8); } while(0)

Definition at line 15 of file structures.h.

Referenced by \_cmd\_macro(), cmd\_notify(), corsair\_mousecopy(), hid\_kb\_translate(), and hid\_mouse\_translate().

8.40.3 Variable Documentation

8.40.3.1 const union devcmd vtable\_keyboard

Definition at line 28 of file device vtable.c.

Referenced by get\_vtable().

8.40.3.2 const union devcmd vtable\_keyboard\_nonrgb

Definition at line 75 of file device\_vtable.c.

Referenced by get\_vtable().

### 8.40.3.3 const union devcmd vtable\_mouse

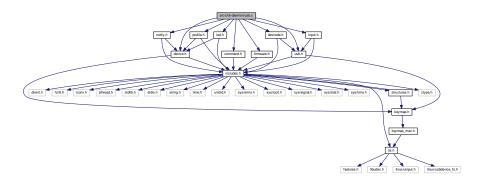
Definition at line 122 of file device\_vtable.c.

Referenced by get vtable().

# 8.41 src/ckb-daemon/usb.c File Reference

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

Include dependency graph for usb.c:



### **Functions**

- const char \* vendor\_str (short vendor)
- const char \* product\_str (short product)
- static const devcmd \* get\_vtable (short vendor, short product)
- static void \* devmain (usbdevice \*kb)
- static void \* \_setupusb (void \*context)
- void setupusb (usbdevice \*kb)
- int revertusb (usbdevice \*kb)
- int \_resetusb (usbdevice \*kb, const char \*file, int line)
- int usb\_tryreset (usbdevice \*kb)
- int \_usbsend (usbdevice \*kb, const uchar \*messages, int count, const char \*file, int line)
- int \_usbrecv (usbdevice \*kb, const uchar \*out\_msg, uchar \*in\_msg, const char \*file, int line)
- int closeusb (usbdevice \*kb)

# **Variables**

- pthread\_mutex\_t usbmutex = PTHREAD\_MUTEX\_INITIALIZER
- volatile int reset stop = 0
- int features\_mask = -1
- · int hwload\_mode

#### 8.41.1 Function Documentation

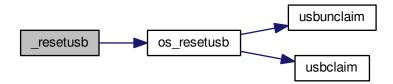
### 8.41.1.1 int\_resetusb ( usbdevice \* kb, const char \* file, int line )

Definition at line 149 of file usb.c.

References usbdevice::active, DELAY\_LONG, os\_resetusb(), and usbdevice::vtable.

```
150
        // Perform a USB reset
        DELAY_LONG(kb);
152
        int res = os_resetusb(kb, file, line);
153
        if(res)
154
            return res:
        DELAY_LONG(kb);
155
156
        // Re-initialize the device
157
        if(kb->vtable->start(kb, kb->active) != 0)
158
            return -1;
159
        if (kb->vtable->updatergb(kb, 1) != 0)
160
            return -1;
        return 0;
161
162 }
```

Here is the call graph for this function:



```
8.41.1.2 static void*_setupusb(void* context) [static]
```

Definition at line 77 of file usb.c.

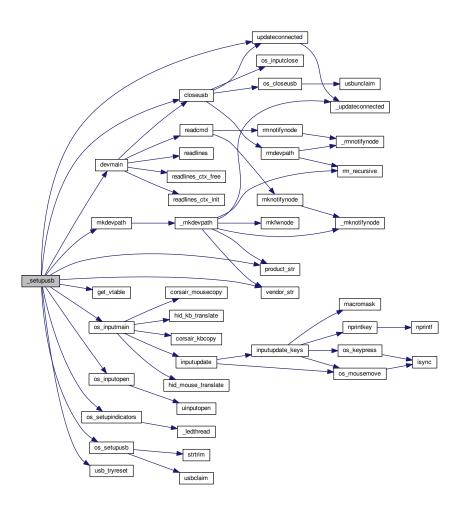
References ckb\_info, closeusb(), DELAY\_LONG, devmain(), devpath, dmutex, FEAT\_ADJRATE, FEAT\_MONOCHROME, FEAT\_STD\_NRGB, FEAT\_STD\_RGB, usbdevice::features, features\_mask, get\_vtable(), imutex, INDEX\_OF, usbdevice::inputthread, IS\_MONOCHROME, IS\_MOUSE, IS\_RGB, KB\_NAME\_LEN, keyboard, mkdevpath(), usbdevice::name, os\_inputmain(), os\_inputopen(), os\_setupindicators(), os\_setupusb(), usbdevice::product, product\_str(), usbdevice::serial, SERIAL\_LEN, updateconnected(), USB\_DELAY\_DEFAULT, usb\_tryreset(), usbdevice::usbdelay, usbdevice::vendor, vendor\_str(), and usbdevice::vtable.

Referenced by setupusb().

```
78
       usbdevice* kb = context;
79
       // Set standard fields
       short vendor = kb->vendor, product = kb->product;
       const devcmd* vt = kb->vtable = get_vtable(vendor, product);
82
       kb->features = (IS_RGB(vendor, product) ? FEAT_STD_RGB :
      FEAT STD NRGB) & features mask:
83
       if(IS_MOUSE(vendor, product)) kb->features |= FEAT_ADJRATE;
       if(IS_MONOCHROME(vendor, product)) kb->features |=
84
      FEAT_MONOCHROME;
85
       kb->usbdelay = USB_DELAY_DEFAULT;
86
87
       // Perform OS-specific setup
88
       DELAY LONG(kb);
       if (os_setupusb(kb))
89
           goto fail;
```

```
// Make up a device name and serial if they weren't assigned
        if(!kb->serial[0])
            snprintf(kb->serial, SERIAL_LEN, "%04x:%04x-NoID", kb->
93
       vendor, kb->product);
if(!kb->name[0])
94
            snprintf(kb->name, KB_NAME_LEN, "%s %s", vendor_str(kb->
95
       vendor), product_str(kb->product));
96
97
        // Set up an input device for key events
        if (os_inputopen(kb))
    goto fail;
98
99
100
         if(pthread_create(&kb->inputthread, 0, os_inputmain, kb))
101
             goto fail;
         pthread_detach(kb->inputthread);
102
103
         if (os_setupindicators(kb))
104
             goto fail;
105
106
         // Set up device
107
         vt->allocprofile(kb);
108
         vt->updateindicators(kb, 1);
109
         pthread_mutex_unlock(imutex(kb));
110
         if(vt->start(kb, 0) && usb_tryreset(kb))
             goto fail_noinput;
111
112
113
         // Make /dev path
114
         if (mkdevpath(kb))
    goto fail_noinput;
115
116
         // Finished. Enter main loop
int index = INDEX_OF(kb, keyboard);
ckb_info("Setup finished for %s%d\n", devpath, index);
117
118
119
120
         updateconnected();
121
         return devmain(kb);
122
123
         fail:
         pthread_mutex_unlock(imutex(kb));
124
125
         fail_noinput:
126
         closeusb(kb);
127
         pthread_mutex_unlock(dmutex(kb));
128
         return 0;
129 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



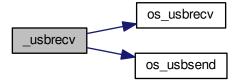
8.41.1.3 int\_usbrecv ( usbdevice \* kb, const uchar \* out\_msg, uchar \* in\_msg, const char \* file, int line )

Definition at line 207 of file usb.c.

References ckb\_err\_fn, DELAY\_LONG, DELAY\_MEDIUM, DELAY\_SHORT, hwload\_mode, os\_usbrecv(), os\_usbsend(), and reset\_stop.

```
216
                // Retry on temporary failure
217
                if(reset_stop)
218
                     return 0;
                DELAY_LONG(kb);
219
220
                continue;
221
            // Wait for the response
222
223
            DELAY_MEDIUM(kb);
224
            res = os_usbrecv(kb, in_msg, file, line);
225
            if(res == 0)
                 return 0;
226
            else if(res != -1)
    return res;
227
228
229
             if(reset_stop || hwload_mode != 2)
230
                return 0;
            DELAY_LONG(kb);
231
232
233
        // Give up
234
        ckb_err_fn("Too many send/recv failures. Dropping.\n", file, line);
235
        return 0;
236 }
```

Here is the call graph for this function:



8.41.1.4 int \_usbsend ( usbdevice \* kb, const uchar \* messages, int count, const char \* file, int line )

Definition at line 184 of file usb.c.

References DELAY\_LONG, DELAY\_SHORT, hwload\_mode, MSG\_SIZE, os\_usbsend(), and reset\_stop.

```
184
                                                                                                             {
185
         int total_sent = 0;
         for(int i = 0; i < count; i++){
    // Send each message via the OS function</pre>
186
187
188
              while(1){
                  DELAY_SHORT (kb);
189
                  int res = os_usbsend(kb, messages + i * MSG_SIZE, 0, file, line);
if(res == 0)
190
191
                      return 0;
192
193
                  else if (res != -1) {
194
                      total_sent += res;
195
                       break;
196
197
                  /// Stop immediately if the program is shutting down or hardware load is set to tryonce
                  if(reset_stop || hwload_mode != 2)
198
199
                       return 0;
                  // Retry as long as the result is temporary failure
DELAY_LONG(kb);
200
201
202
              }
203
204
         return total_sent;
205 }
```

Here is the call graph for this function:



### 8.41.1.5 int closeusb ( usbdevice \* kb )

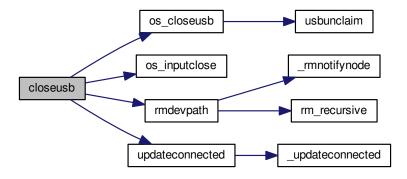
Definition at line 238 of file usb.c.

References ckb\_info, devpath, dmutex, usbdevice::handle, imutex, INDEX\_OF, keyboard, os\_closeusb(), os\_inputclose(), rmdevpath(), usbdevice::thread, updateconnected(), and usbdevice::vtable.

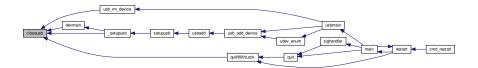
Referenced by \_setupusb(), devmain(), quitWithLock(), and usb\_rm\_device().

```
238
239
        pthread_mutex_lock(imutex(kb));
240
        if (kb->handle) {
             int index = INDEX_OF(kb, keyboard);
ckb_info("Disconnecting %s%d\n", devpath, index);
241
242
243
             os_inputclose(kb);
244
             updateconnected();
             // Close USB device
245
246
             os_closeusb(kb);
247
        } else
248
             updateconnected();
249
        rmdevpath(kb);
250
251
        // Wait for thread to close
        pthread_mutex_unlock(imutex(kb));
252
253
        pthread_mutex_unlock(dmutex(kb));
254
        pthread_join(kb->thread, 0);
255
        pthread_mutex_lock(dmutex(kb));
256
257
        // Delete the profile and the control path
258
        if(!kb->vtable)
259
             return 0;
260
        kb->vtable->freeprofile(kb);
261
        memset(kb, 0, sizeof(usbdevice));
2.62
        return 0;
263 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



```
8.41.1.6 static void* devmain ( usbdevice * kb ) [static]
```

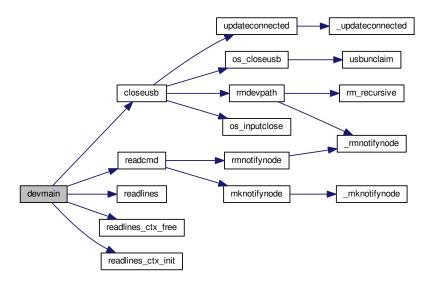
Definition at line 50 of file usb.c.

References closeusb(), dmutex, usbdevice::infifo, IS\_CONNECTED, readcmd(), readlines(), readlines\_ctx\_free(), and readlines\_ctx\_init().

Referenced by \_setupusb().

```
// dmutex should still be locked when this is called
52
        int kbfifo = kb->infifo - 1;
        readlines_ctx linectx;
53
        readlines_ctx_init(&linectx);
54
        while(1){
55
             pthread_mutex_unlock(dmutex(kb));
56
57
             // Read from FIFO
58
             const char* line;
             int lines = readlines(kbfifo, linectx, &line);
pthread_mutex_lock(dmutex(kb));
// End thread when the handle is removed
59
60
61
             if(!IS_CONNECTED(kb))
62
             if(lines){
65
                  if(readcmd(kb, line)){
                       // USB transfer failed; destroy device
closeusb(kb);
66
67
68
                       break;
69
70
71
72
73
        {\tt pthread\_mutex\_unlock\,(dmutex\,(kb)\,)\,;}\\
        readlines_ctx_free(linectx);
74
        return 0;
```

Here is the call graph for this function:



Here is the caller graph for this function:



8.41.1.7 static const devcmd\* get\_vtable ( short vendor, short product ) [static]

Definition at line 45 of file usb.c.

References IS\_MOUSE, IS\_RGB, vtable\_keyboard, vtable\_keyboard\_nonrgb, and vtable\_mouse.

Referenced by \_setupusb().

```
45
46

return IS_MOUSE(vendor, product) ? &vtable_mouse :
IS_RGB(vendor, product) ? &vtable_keyboard : &
vtable_keyboard_nonrgb;
47 }
```

Here is the caller graph for this function:



8.41.1.8 const char\* product\_str ( short product )

Definition at line 26 of file usb.c.

References P\_K65, P\_K65\_LUX, P\_K65\_NRGB, P\_K65\_RFIRE, P\_K70, P\_K70\_LUX, P\_K70\_LUX\_NRGB, P\_K70\_NRGB, P\_K70\_NRGB, P\_K70\_RFIRE, P\_K70\_RFIRE\_NRGB, P\_K95, P\_K95\_NRGB, P\_K95\_PLATINUM, P\_M65, P\_M65\_PRO, P\_SABRE\_L, P\_SABRE\_N, P\_SABRE\_O, P\_SABRE\_O2, P\_SCIMITAR, P\_SCIMITAR\_PRO, P\_STRAFE, and P\_STRAFE\_NRGB.

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

```
if(product == P_K95 || product == P_K95_NRGB || product ==
27
       P_K95_PLATINUM)
            return "k95";
28
      if(product == P_K70 || product == P_K70_NRGB || product ==
P_K70_LUX || product == P_K70_LUX_NRGB || product ==
29
       P_K70_RFIRE || product == P_K70_RFIRE_NRGB)
      return "k70";

if(product == P_K65 || product == P_K65_NRGB || product ==

P_K65_LUX || product == P_K65_RFIRE)
30
31
            return "k65";
32
        if(product == P_STRAFE || product == P_STRAFE_NRGB)
33
            return "strafe";
35
        if (product == P_M65 || product == P_M65_PRO)
       return "m65";
if(product == P_SABRE_L || product ==
36
37
       P_SABRE_N || product == P_SABRE_02)
38
            return "sabre";
        if(product == P_SCIMITAR || product == P_SCIMITAR_PRO)
39
       return "scimitar";
return "";
40
41
42 }
```

Here is the caller graph for this function:



### 8.41.1.9 int revertusb ( usbdevice \* kb )

Definition at line 137 of file usb.c.

References FEAT RGB, HAS FEATURES, NEEDS FW UPDATE, NK95 HWON, nk95cmd, and setactive.

Referenced by quitWithLock().

```
137
138
        if (NEEDS_FW_UPDATE(kb))
139
            return 0;
140
        if(!HAS_FEATURES(kb, FEAT_RGB)){
141
            nk95cmd(kb, NK95_HWON);
142
            return 0;
143
144
        if (setactive(kb, 0))
145
            return -1;
        return 0;
146
147 }
```

Here is the caller graph for this function:



### 8.41.1.10 void setupusb ( usbdevice \*kb )

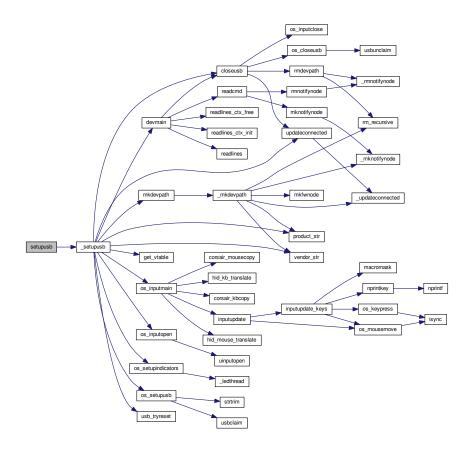
Definition at line 131 of file usb.c.

References \_setupusb(), ckb\_err, imutex, and usbdevice::thread.

Referenced by usbadd().

```
131 {
132 pthread_mutex_lock(imutex(kb));
133 if(pthread_create(&kb->thread, 0, _setupusb, kb))
134 ckb_err("Failed to create USB thread\n");
135 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



# 8.41.1.11 int usb\_tryreset ( usbdevice \* kb )

Definition at line 164 of file usb.c.

References ckb\_err, ckb\_info, reset\_stop, and resetusb.

Referenced by \_setupusb(), and cmd\_fwupdate().

```
164
165
        if(reset_stop)
166
            return -1;
        ckb_info("Attempting reset...\n");
167
168
       while(1){
169
            int res = resetusb(kb);
170
            if(!res){
                ckb_info("Reset success\n");
171
172
                return 0;
173
174
            if(res == -2 || reset_stop)
175
176
177
        ckb_err("Reset failed. Disconnecting.\n");
178
        return -1;
179 }
```

Here is the caller graph for this function:



### 8.41.1.12 const char\* vendor\_str ( short vendor )

Definition at line 20 of file usb.c.

References V\_CORSAIR.

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

```
20
21    if(vendor == V_CORSAIR)
22        return "corsair";
23    return "";
24 }
```

Here is the caller graph for this function:



### 8.41.2 Variable Documentation

8.41.2.1 int features\_mask = -1

Definition at line 17 of file usb.c.

Referenced by \_setupusb(), and main().

8.41.2.2 int hwload\_mode

Definition at line 7 of file device.c.

Referenced by \_start\_dev(), \_usbrecv(), and \_usbsend().

8.41.2.3 volatile int reset\_stop = 0

Definition at line 14 of file usb.c.

Referenced by \_usbrecv(), \_usbsend(), quitWithLock(), and usb\_tryreset().

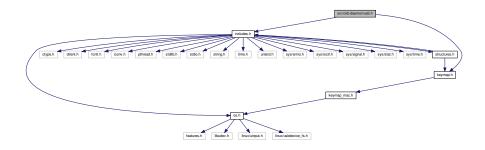
8.41.2.4 pthread\_mutex\_t usbmutex = PTHREAD\_MUTEX\_INITIALIZER

Definition at line 11 of file usb.c.

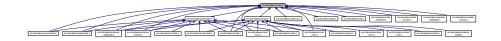
# 8.42 src/ckb-daemon/usb.h File Reference

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

Include dependency graph for usb.h:



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



### **Macros**

- #define V\_CORSAIR 0x1b1c
- #define V\_CORSAIR\_STR "1b1c"
- #define P K65 0x1b17
- #define P\_K65\_STR "1b17"
- #define P\_K65\_NRGB 0x1b07
- #define P\_K65\_NRGB\_STR "1b07"
- #define P\_K65\_LUX 0x1b37
- #define P K65 LUX STR "1b37"
- #define P\_K65\_RFIRE 0x1b39
- #define P\_K65\_RFIRE\_STR "1b39"
- #define IS\_K65(kb) ((kb)->vendor == V\_CORSAIR && ((kb)->product == P\_K65 || (kb)->product == P\_K65\_LUX || (kb)->product == P\_K65\_RFIRE))
- #define P\_K70 0x1b13
- #define P K70 STR "1b13"
- #define P\_K70\_NRGB 0x1b09
- #define P K70 NRGB STR "1b09"
- #define P\_K70\_LUX 0x1b33

- #define P\_K70\_LUX\_STR "1b33"
- #define P\_K70\_LUX\_NRGB 0x1b36
- #define P\_K70\_LUX\_NRGB\_STR "1b36"
- #define P K70 RFIRE 0x1b38
- #define P K70 RFIRE STR "1b38"
- #define P\_K70\_RFIRE\_NRGB 0x1b3a
- #define P K70 RFIRE NRGB STR "1b3a"
- #define IS\_K70(kb) ((kb)->vendor == V\_CORSAIR && ((kb)->product == P\_K70 || (kb)->product == P\_K70\_NRGB || (kb)->product == P\_K70\_RFIRE || (kb)->product == P\_K70\_LUX || (kb)->product == P\_K70\_LUX\_NRGB))
- #define P K95 0x1b11
- #define P\_K95\_STR "1b11"
- #define P\_K95\_NRGB 0x1b08
- #define P K95 NRGB STR "1b08"
- #define P K95 PLATINUM 0x1b2d
- #define P K95 PLATINUM STR "1b2d"
- #define IS\_K95(kb) ((kb)->vendor == V\_CORSAIR && ((kb)->product == P\_K95 || (kb)->product == P\_K95-NRGB || (kb)->product == P\_K95\_PLATINUM))
- #define P STRAFE 0x1b20
- #define P STRAFE STR "1b20"
- #define P STRAFE NRGB 0x1b15
- #define P STRAFE NRGB STR "1b15"
- #define IS\_STRAFE(kb) ((kb)->vendor == V\_CORSAIR && ((kb)->product == P\_STRAFE || (kb)->product == P\_STRAFE\_NRGB))
- #define P\_M65 0x1b12
- #define P M65 STR "1b12"
- #define P\_M65\_PRO 0x1b2e
- #define P\_M65\_PRO\_STR "1b2e"
- #define IS\_M65(kb) ((kb)->vendor == V\_CORSAIR && ((kb)->product == P\_M65 || (kb)->product == P\_M65\_PRO))
- #define P\_SABRE\_O 0x1b14 /\* optical \*/
- #define P\_SABRE\_O\_STR "1b14"
- #define P\_SABRE\_L 0x1b19 /\* laser \*/
- #define P\_SABRE\_L\_STR "1b19"
- #define P\_SABRE\_N 0x1b2f /\* new? \*/
- #define P\_SABRE\_N\_STR "1b2f"
- #define P\_SABRE\_O2 0x1b32 /\* Observed on a CH-9000111-EU model SABRE \*/
- #define P\_SABRE\_O2\_STR "1b32"
- #define IS\_SABRE(kb) ((kb)->vendor == V\_CORSAIR && ((kb)->product == P\_SABRE\_O || (kb)->product == P\_SABRE\_L || (kb)->product == P\_SABRE\_N || (kb)->product == P\_SABRE\_O2))
- #define P SCIMITAR 0x1b1e
- #define P\_SCIMITAR\_STR "1b1e"
- #define P\_SCIMITAR\_PRO 0x1b3e
- #define P SCIMITAR PRO STR "1b3e"
- #define IS\_SCIMITAR(kb) ((kb)->vendor == V\_CORSAIR && ((kb)->product == P\_SCIMITAR || (kb)->product == P\_SCIMITAR\_PRO))
- #define IS\_RGB(vendor, product) ((vendor) == (V\_CORSAIR) && (product) != (P\_K65\_NRGB) && (product) != (P\_K70\_NRGB) && (product) != (P\_K95\_NRGB))
- #define IS\_MONOCHROME(vendor, product) ((vendor) == (V\_CORSAIR) && (product) == (P\_STRAFE\_N-RGB))
- #define IS\_RGB\_DEV(kb) IS\_RGB((kb)->vendor, (kb)->product)
- #define IS\_MONOCHROME\_DEV(kb) IS\_MONOCHROME((kb)->vendor, (kb)->product)
- #define IS\_FULLRANGE(kb) (IS\_RGB((kb)->vendor, (kb)->product) && (kb)->product != P\_K65 && (kb)->product != P\_K70 && (kb)->product != P\_K95)

- #define IS\_MOUSE(vendor, product) ((vendor) == (V\_CORSAIR) && ((product) == (P\_M65) || (product) == (P\_M65\_PRO) || (product) == (P\_SABRE\_O) || (product) == (P\_SABRE\_L) || (product) == (P\_SCIMITAR) || (product) == (P\_SCIMITAR\_PRO) || (product) == (P\_SABRE\_O2)))
- #define IS\_MOUSE\_DEV(kb) IS\_MOUSE((kb)->vendor, (kb)->product)
- #define DELAY\_SHORT(kb) usleep((int)(kb)->usbdelay \* 1000)
- #define DELAY MEDIUM(kb) usleep((int)(kb)->usbdelay \* 10000)
- #define DELAY\_LONG(kb) usleep(100000)
- #define USB DELAY DEFAULT 5
- #define resetusb(kb) \_resetusb(kb, \_\_FILE\_NOPATH\_\_, \_\_LINE\_\_)
- #define usbsend(kb, messages, count) \_usbsend(kb, messages, count, \_\_FILE\_NOPATH\_\_, \_\_LINE\_\_)
- #define usbrecv(kb, out\_msg, in\_msg) \_usbrecv(kb, out\_msg, in\_msg, \_\_FILE\_NOPATH\_\_, \_\_LINE\_\_)
- #define nk95cmd(kb, command) \_nk95cmd(kb, (command) >> 16 & 0xFF, (command) & 0xFFFF, \_\_FILE\_-NOPATH\_\_, \_LINE\_\_)
- #define NK95 HWOFF 0x020030
- #define NK95\_HWON 0x020001
- #define NK95 M1 0x140001
- #define NK95\_M2 0x140002
- #define NK95 M3 0x140003

#### **Functions**

- const char \* vendor\_str (short vendor)
- const char \* product str (short product)
- int usbmain ()
- void usbkill ()
- void setupusb (usbdevice \*kb)
- int os setupusb (usbdevice \*kb)
- void \* os\_inputmain (void \*kb)
- int revertusb (usbdevice \*kb)
- int closeusb (usbdevice \*kb)
- void os\_closeusb (usbdevice \*kb)
   int resetusb (usbdevice \*kb, const char \*file, int line)
- int os resetusb (usbdevice \*kb, const char \*file, int line)
- int os\_resetusb (usbdevice \*kb, const char \*file, int line)
   int \_usbsend (usbdevice \*kb, const uchar \*messages, int count, const char \*file, int line)
- int \_usbrecv (usbdevice \*kb, const uchar \*out\_msg, uchar \*in\_msg, const char \*file, int line)
- int os\_usbsend (usbdevice \*kb, const uchar \*out\_msg, int is\_recv, const char \*file, int line)
- int os\_usbrecv (usbdevice \*kb, uchar \*in\_msg, const char \*file, int line)
- void os sendindicators (usbdevice \*kb)
- int nk95cmd (usbdevice \*kb, uchar bRequest, ushort wValue, const char \*file, int line)
- int usb\_tryreset (usbdevice \*kb)

# 8.42.1 Macro Definition Documentation

### 8.42.1.1 #define DELAY\_LONG( kb ) usleep(100000)

Definition at line 100 of file usb.h.

Referenced by \_resetusb(), \_usbrecv(), \_usbsend(), cmd\_hwload\_kb(), cmd\_hwload\_mouse(), cmd\_hwsave\_kb(), and cmd\_hwsave\_mouse().

8.42.1.2 #define DELAY\_MEDIUM( kb ) usleep((int)(kb)->usbdelay \* 10000)

Definition at line 99 of file usb.h.

Referenced by \_usbrecv(), and setactive\_kb().

8.42.1.3 #define DELAY\_SHORT( kb ) usleep((int)(kb)->usbdelay \* 1000)

Definition at line 98 of file usb.h.

Referenced by usbrecv(), usbsend(), and updateindicators kb().

8.42.1.4 #define IS\_FULLRANGE( kb ) (IS\_RGB((kb)->vendor, (kb)->product) && (kb)->product != P\_K65 && (kb)->product != P\_K70 && (kb)->product != P\_K95)

Definition at line 91 of file usb.h.

Referenced by readcmd(), and updatergb kb().

8.42.1.5 #define IS\_K65( kb ) ((kb)->vendor == V\_CORSAIR && ((kb)->product == P\_K65 || (kb)->product == P\_K65\_NRGB || (kb)->product == P\_K65\_LUX || (kb)->product == P\_K65\_RFIRE))

Definition at line 19 of file usb.h.

Referenced by has\_key().

8.42.1.6 #define IS\_K70( kb ) ((kb)->vendor == V\_CORSAIR && ((kb)->product == P\_K70 || (kb)->product == P\_K70\_NRGB || (kb)->product == P\_K70\_RFIRE || (kb)->product == P\_K70\_LUX || (kb)->product == P\_K70\_LUX\_NRGB))

Definition at line 33 of file usb.h.

8.42.1.7 #define IS\_K95( kb ) ((kb)->vendor == V\_CORSAIR && ((kb)->product == P\_K95 || (kb)->product == P\_K95 || (kb)->

Definition at line 41 of file usb.h.

Referenced by cmd\_hwload\_kb(), cmd\_hwsave\_kb(), and has\_key().

8.42.1.8 #define IS\_M65( kb ) ((kb)->vendor == V\_CORSAIR && ((kb)->product == P\_M65 || (kb)->product == P\_M65 PRO))

Definition at line 53 of file usb.h.

Referenced by isblack().

8.42.1.9 #define IS\_MONOCHROME( vendor, product ) ((vendor) == (V CORSAIR) && (product) == (P STRAFE NRGB))

Definition at line 86 of file usb.h.

Referenced by \_setupusb().

8.42.1.10 #define IS\_MONOCHROME\_DEV( kb ) IS\_MONOCHROME((kb)->vendor, (kb)->product)

Definition at line 88 of file usb.h.

8.42.1.11 #define IS\_MOUSE( vendor, product ) ((vendor) == (V\_CORSAIR) && ((product) == (P\_M65) || (product) == (P\_SABRE\_N) || (product) == (P\_SABRE\_N) || (product) == (P\_SCIMITAR) || (product) == (P\_SCIMITAR PRO) || (product) == (P\_SABRE\_O2)))

Definition at line 94 of file usb.h.

Referenced by \_setupusb(), get\_vtable(), has\_key(), and os\_inputmain().

8.42.1.12 #define IS\_MOUSE\_DEV( kb ) IS\_MOUSE((kb)->vendor, (kb)->product)

Definition at line 95 of file usb.h.

Referenced by readcmd().

8.42.1.13 #define IS\_RGB( vendor, product ) ((vendor) == (V\_CORSAIR) && (product) != (P\_K65\_NRGB) && (product) != (P\_K70\_NRGB) && (product) != (P\_K95\_NRGB))

Definition at line 85 of file usb.h.

Referenced by \_setupusb(), get\_vtable(), and os\_inputmain().

8.42.1.14 #define IS\_RGB\_DEV( kb ) IS\_RGB((kb)->vendor, (kb)->product)

Definition at line 87 of file usb.h.

8.42.1.15 #define IS\_SABRE( kb ) ((kb)->vendor == V\_CORSAIR && ((kb)->product == P\_SABRE\_O || (kb)->product == P\_SABRE\_O2))

Definition at line 63 of file usb.h.

Referenced by has\_key(), loadrgb\_mouse(), and savergb\_mouse().

8.42.1.16 #define IS\_SCIMITAR( kb ) ((kb)->vendor == V\_CORSAIR && ((kb)->product == P\_SCIMITAR || (kb)->product == P\_SCIMITAR\_PRO))

Definition at line 69 of file usb.h.

Referenced by has\_key(), loadrgb\_mouse(), and savergb\_mouse().

8.42.1.17 #define IS\_STRAFE( kb ) ((kb)->vendor == V\_CORSAIR && ((kb)->product == P\_STRAFE || (kb)->product == P\_STRAFE\_NRGB))

Definition at line 47 of file usb.h.

Referenced by savergb\_kb().

8.42.1.18 #define NK95\_HWOFF 0x020030

Definition at line 146 of file usb.h.

Referenced by start kb nrgb().

8.42.1.19 #define NK95 HWON 0x020001

Definition at line 147 of file usb.h.

Referenced by revertusb().

```
8.42.1.20 #define NK95_M1 0x140001
Definition at line 148 of file usb.h.
Referenced by setmodeindex_nrgb().
8.42.1.21 #define NK95_M2 0x140002
Definition at line 149 of file usb.h.
Referenced by setmodeindex_nrgb().
8.42.1.22 #define NK95_M3 0x140003
Definition at line 150 of file usb.h.
Referenced by setmodeindex nrgb().
8.42.1.23 #define nk95cmd( \it kb, command ) \_nk95cmd(kb, (command) >> 16 & 0xFF, (command) & 0xFFFF,
          __FILE_NOPATH__, __LINE__)
Definition at line 144 of file usb.h.
Referenced by revertusb(), setmodeindex_nrgb(), and start_kb_nrgb().
8.42.1.24 #define P_K65 0x1b17
Definition at line 11 of file usb.h.
Referenced by product_str().
8.42.1.25 #define P_K65_LUX 0x1b37
Definition at line 15 of file usb.h.
Referenced by product_str().
8.42.1.26 #define P_K65_LUX_STR "1b37"
Definition at line 16 of file usb.h.
8.42.1.27 #define P_K65_NRGB 0x1b07
Definition at line 13 of file usb.h.
Referenced by product_str().
8.42.1.28 #define P_K65_NRGB_STR "1b07"
Definition at line 14 of file usb.h.
8.42.1.29 #define P_K65_RFIRE 0x1b39
Definition at line 17 of file usb.h.
```

Referenced by product\_str().

8.42.1.30 #define P\_K65\_RFIRE\_STR "1b39"

Definition at line 18 of file usb.h.

8.42.1.31 #define P\_K65\_STR "1b17"

Definition at line 12 of file usb.h.

8.42.1.32 #define P\_K70 0x1b13

Definition at line 21 of file usb.h.

Referenced by product\_str().

8.42.1.33 #define P\_K70\_LUX 0x1b33

Definition at line 25 of file usb.h.

Referenced by product\_str().

8.42.1.34 #define P\_K70\_LUX\_NRGB 0x1b36

Definition at line 27 of file usb.h.

Referenced by product\_str().

8.42.1.35 #define P\_K70\_LUX\_NRGB\_STR "1b36"

Definition at line 28 of file usb.h.

8.42.1.36 #define P\_K70\_LUX\_STR "1b33"

Definition at line 26 of file usb.h.

8.42.1.37 #define P\_K70\_NRGB 0x1b09

Definition at line 23 of file usb.h.

Referenced by product\_str().

8.42.1.38 #define P\_K70\_NRGB\_STR "1b09"

Definition at line 24 of file usb.h.

8.42.1.39 #define P\_K70\_RFIRE 0x1b38

Definition at line 29 of file usb.h.

Referenced by product\_str().

8.42.1.40 #define P\_K70\_RFIRE\_NRGB 0x1b3a

Definition at line 31 of file usb.h.

Referenced by product\_str().

8.42.1.41 #define P\_K70\_RFIRE\_NRGB\_STR "1b3a"

Definition at line 32 of file usb.h.

8.42.1.42 #define P\_K70\_RFIRE\_STR "1b38"

Definition at line 30 of file usb.h.

8.42.1.43 #define P\_K70\_STR "1b13"

Definition at line 22 of file usb.h.

8.42.1.44 #define P\_K95 0x1b11

Definition at line 35 of file usb.h.

Referenced by product\_str().

8.42.1.45 #define P\_K95\_NRGB 0x1b08

Definition at line 37 of file usb.h.

Referenced by \_nk95cmd(), and product\_str().

8.42.1.46 #define P\_K95\_NRGB\_STR "1b08"

Definition at line 38 of file usb.h.

8.42.1.47 #define P\_K95\_PLATINUM 0x1b2d

Definition at line 39 of file usb.h.

Referenced by product\_str().

8.42.1.48 #define P\_K95\_PLATINUM\_STR "1b2d"

Definition at line 40 of file usb.h.

8.42.1.49 #define P\_K95\_STR "1b11"

Definition at line 36 of file usb.h.

8.42.1.50 #define P\_M65 0x1b12

Definition at line 49 of file usb.h.

Referenced by product\_str().

```
8.42.1.51 #define P_M65_PRO 0x1b2e
Definition at line 51 of file usb.h.
Referenced by product_str().
8.42.1.52 #define P_M65_PRO_STR "1b2e"
Definition at line 52 of file usb.h.
8.42.1.53 #define P_M65_STR "1b12"
Definition at line 50 of file usb.h.
8.42.1.54 #define P_SABRE_L 0x1b19 /* laser */
Definition at line 57 of file usb.h.
Referenced by product_str().
8.42.1.55 #define P_SABRE_L_STR "1b19"
Definition at line 58 of file usb.h.
8.42.1.56 #define P_SABRE_N 0x1b2f /* new? */
Definition at line 59 of file usb.h.
Referenced by product_str().
8.42.1.57 #define P SABRE N STR "1b2f"
Definition at line 60 of file usb.h.
8.42.1.58 #define P_SABRE_O 0x1b14 /* optical */
Definition at line 55 of file usb.h.
Referenced by product_str().
8.42.1.59 #define P_SABRE_02 0x1b32 /* Observed on a CH-9000111-EU model SABRE */
Definition at line 61 of file usb.h.
Referenced by product_str().
8.42.1.60 #define P_SABRE_O2_STR "1b32"
Definition at line 62 of file usb.h.
8.42.1.61 #define P_SABRE_O_STR "1b14"
```

Definition at line 56 of file usb.h.

8.42.1.62 #define P\_SCIMITAR 0x1b1e Definition at line 65 of file usb.h. Referenced by product\_str(). 8.42.1.63 #define P\_SCIMITAR\_PRO 0x1b3e Definition at line 67 of file usb.h. Referenced by product\_str(). 8.42.1.64 #define P\_SCIMITAR\_PRO\_STR "1b3e" Definition at line 68 of file usb.h. 8.42.1.65 #define P\_SCIMITAR\_STR "1b1e" Definition at line 66 of file usb.h. 8.42.1.66 #define P\_STRAFE 0x1b20 Definition at line 43 of file usb.h. Referenced by product\_str(). 8.42.1.67 #define P\_STRAFE\_NRGB 0x1b15 Definition at line 45 of file usb.h. Referenced by product\_str(). 8.42.1.68 #define P\_STRAFE\_NRGB\_STR "1b15" Definition at line 46 of file usb.h. 8.42.1.69 #define P\_STRAFE\_STR "1b20" Definition at line 44 of file usb.h. 8.42.1.70 #define resetusb( kb ) \_resetusb(kb, \_\_FILE\_NOPATH\_\_, \_\_LINE\_\_) Definition at line 125 of file usb.h. Referenced by usb\_tryreset(). 8.42.1.71 #define USB\_DELAY\_DEFAULT 5 Definition at line 101 of file usb.h.

Referenced by \_setupusb(), and start\_dev().

```
8.42.1.72 #define usbrecv( kb, out_msg, in_msg ) usbrecv(kb, out_msg, in_msg, __FILE_NOPATH__, _LINE__)
```

Definition at line 133 of file usb.h.

Referenced by cmd\_hwload\_kb(), cmd\_hwload\_mouse(), getfwversion(), hwloadmode(), loaddpi(), loaddpi(), and loadrgb\_mouse().

```
8.42.1.73 #define usbsend( kb, messages, count ) _usbsend(kb, messages, count, __FILE_NOPATH__, _LINE__)
```

Definition at line 130 of file usb.h.

Referenced by cmd\_hwsave\_kb(), cmd\_hwsave\_mouse(), cmd\_pollrate(), fwupdate(), loadrgb\_kb(), savedpi(), savergb\_kb(), savergb\_mouse(), setactive\_mouse(), updatedpi(), updatergb\_kb(), and updatergb\_mouse().

#### 8.42.1.74 #define V\_CORSAIR 0x1b1c

Definition at line 8 of file usb.h.

Referenced by usb\_add\_device(), and vendor\_str().

```
8.42.1.75 #define V_CORSAIR_STR "1b1c"
```

Definition at line 9 of file usb.h.

Referenced by udev enum(), and usb add device().

#### 8.42.2 Function Documentation

```
8.42.2.1 int_nk95cmd ( usbdevice * kb, uchar bRequest, ushort wValue, const char * file, int line )
```

Definition at line 74 of file usb\_linux.c.

References ckb\_err\_fn, usbdevice::handle, P\_K95\_NRGB, and usbdevice::product.

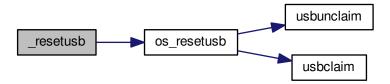
### 8.42.2.2 int resetusb ( usbdevice \*kb, const char \*file, int line )

Definition at line 149 of file usb.c.

References usbdevice::active, DELAY\_LONG, os\_resetusb(), and usbdevice::vtable.

```
149
                                                                  {
        // Perform a USB reset
150
151
        DELAY_LONG(kb);
        int res = os_resetusb(kb, file, line);
153
        if(res)
        return res;
DELAY_LONG(kb);
154
155
156
        // Re-initialize the device
157
        if (kb->vtable->start(kb, kb->active) != 0)
158
             return -1;
```

Here is the call graph for this function:



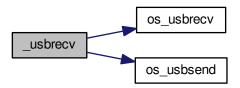
8.42.2.3 int\_usbrecv ( usbdevice \*kb, const uchar  $*out\_msg$ , uchar  $*in\_msg$ , const char \*file, int line )

Definition at line 207 of file usb.c.

References ckb\_err\_fn, DELAY\_LONG, DELAY\_MEDIUM, DELAY\_SHORT, hwload\_mode, os\_usbrecv(), os\_usbsend(), and reset\_stop.

```
207
          // Try a maximum of 3 times
for(int try = 0; try < 5; try++) {
    // Send the output message
    DELAY_SHORT(kb);</pre>
208
209
210
211
              int res = os_usbsend(kb, out_msg, 1, file, line);
if(res == 0)
213
               return 0;
else if (res == -1) {
214
215
                   // Retry on temporary failure
216
                    if(reset_stop)
218
                         return 0;
                    DELAY_LONG(kb);
219
220
                    continue;
221
               // Wait for the response
222
              DELAY_MEDIUM(kb);
223
224
               res = os_usbrecv(kb, in_msg, file, line);
225
               if(res == 0)
                    return 0;
226
               else if(res != -1)
    return res;
227
228
229
               if(reset_stop || hwload_mode != 2)
230
                    return 0;
231
               DELAY_LONG(kb);
232
          // Give up
ckb_err_fn("Too many send/recv failures. Dropping.\n", file, line);
233
234
235
          return 0;
236 }
```

Here is the call graph for this function:



8.42.2.4 int \_usbsend ( usbdevice \* kb, const uchar \* messages, int count, const char \* file, int line )

Definition at line 184 of file usb.c.

References DELAY\_LONG, DELAY\_SHORT, hwload\_mode, MSG\_SIZE, os\_usbsend(), and reset\_stop.

```
185
         int total_sent = 0;
186
        for (int i = 0; i < count; i++) {</pre>
187
             \ensuremath{//} Send each message via the OS function
             while(1){
188
                 DELAY_SHORT (kb);
189
                 int res = os_usbsend(kb, messages + i * MSG_SIZE, 0, file, line);
190
191
                 if(res == 0)
192
                      return 0;
193
                 else if(res != -1){
194
                      total_sent += res;
195
                      break:
196
                 /// Stop immediately if the program is shutting down or hardware load is set to tryonce
197
198
                 if(reset_stop || hwload_mode != 2)
199
                      return 0;
                 \slash\hspace{-0.4em} // Retry as long as the result is temporary failure
200
201
                 DELAY_LONG(kb);
202
             }
204
        return total_sent;
205 }
```

Here is the call graph for this function:



### 8.42.2.5 int closeusb ( usbdevice \* kb )

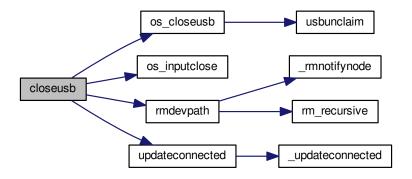
Definition at line 238 of file usb.c.

References ckb\_info, devpath, dmutex, usbdevice::handle, imutex, INDEX\_OF, keyboard, os\_closeusb(), os\_inputclose(), rmdevpath(), usbdevice::thread, updateconnected(), and usbdevice::vtable.

Referenced by \_setupusb(), devmain(), quitWithLock(), and usb\_rm\_device().

```
pthread_mutex_lock(imutex(kb));
239
240
         if(kb->handle){
            int index = INDEX_OF(kb, keyboard);
ckb_info("Disconnecting %s%d\n", devpath, index);
241
242
243
            os_inputclose(kb);
244
            updateconnected();
245
             // Close USB device
246
             os_closeusb(kb);
247
        } else
            updateconnected();
248
        rmdevpath(kb);
249
251
         // Wait for thread to close
252
        pthread_mutex_unlock(imutex(kb));
253
        pthread_mutex_unlock(dmutex(kb));
        pthread_join(kb->thread, 0);
254
255
        pthread_mutex_lock(dmutex(kb));
256
        // Delete the profile and the control path
258
        if(!kb->vtable)
259
             return 0;
260
        kb->vtable->freeprofile(kb);
261
        memset(kb, 0, sizeof(usbdevice));
262
        return 0;
263 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



8.42.2.6 void os\_closeusb ( usbdevice \* kb )

Definition at line 214 of file usb\_linux.c.

References usbdevice::handle, INDEX\_OF, kbsyspath, keyboard, usbdevice::udev, and usbunclaim().

Referenced by closeusb().

```
214
                                      {
215
        if (kb->handle) {
216
            usbunclaim(kb, 0);
217
            close(kb->handle - 1);
218
        if (kb->udev)
219
            udev_device_unref(kb->udev);
220
221
        kb->handle = 0;
222
        kb->udev = 0;
223
        kbsyspath[INDEX_OF(kb, keyboard)][0] = 0;
224 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



#### 8.42.2.7 void\* os\_inputmain ( void \* kb )

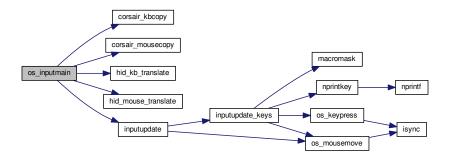
Definition at line 93 of file usb linux.c.

References usbdevice::active, ckb\_info, corsair\_kbcopy(), corsair\_mousecopy(), devpath, usbdevice::epcount, usbdevice::handle, hid\_kb\_translate(), hid\_mouse\_translate(), imutex, INDEX\_OF, usbdevice::input, inputupdate(), IS\_MOUSE, IS\_RGB, keyboard, usbinput::keys, MSG\_SIZE, usbdevice::product, usbinput::rel\_x, usbinput::rel\_y, and usbdevice::vendor.

Referenced by \_setupusb().

```
93
94
       usbdevice* kb = context;
       int fd = kb->handle - 1;
95
       short vendor = kb->vendor, product = kb->product;
96
97
       int index = INDEX_OF(kb, keyboard);
98
       ckb_info("Starting input thread for %s%d\n", devpath, index);
99
100
         // Monitor input transfers on all endpoints for non-RGB devices
        // For RGB, monitor all but the last, as it's used for input/output
101
        int urbcount = IS_RGB(vendor, product) ? (kb->epcount - 1) : kb->
102
      epcount;
103
        struct usbdevfs_urb urbs[urbcount];
104
        memset(urbs, 0, sizeof(urbs));
105
        urbs[0].buffer_length = 8;
        if(IS_RGB(vendor, product)) {
    if(IS_MOUSE(vendor, product))
        urbs[1].buffer_length = 10;
106
107
108
109
110
                 urbs[1].buffer_length = 21;
111
             urbs[2].buffer_length = MSG_SIZE;
112
             if (urbcount != 3)
113
                 urbs[urbcount - 1].buffer_length = MSG_SIZE;
114
```

```
115
             urbs[1].buffer_length = 4;
116
             urbs[2].buffer_length = 15;
117
         // Submit URBs
118
         for (int i = 0; i < urbcount; i++) {
119
             urbs[i].type = USBDEVFS_URB_TYPE_INTERRUPT;
urbs[i].endpoint = 0x80 | (i + 1);
120
121
122
             urbs[i].buffer = malloc(urbs[i].buffer_length);
123
             ioctl(fd, USBDEVFS_SUBMITURB, urbs + i);
124
         // Start monitoring input
125
126
        while(1){
127
             struct usbdevfs_urb* urb = 0;
128
             if(ioctl(fd, USBDEVFS_REAPURB, &urb)){
129
                  if(errno == ENODEV || errno == ENOENT || errno == ESHUTDOWN)
130
                      // Stop the thread if the handle closes
131
                      break:
                 else if(errno == EPIPE && urb){
    // On EPIPE, clear halt on the endpoint
132
133
134
                      ioctl(fd, USBDEVFS_CLEAR_HALT, &urb->endpoint);
135
                      // Re-submit the URB
136
                      if(urb)
                          ioctl(fd, USBDEVFS SUBMITURB, urb);
137
                      urb = 0;
138
139
                 }
140
141
             if(urb){
142
                  // Process input (if any)
                 pthread_mutex_lock(imutex(kb));
if(IS_MOUSE(vendor, product)){
143
144
145
                      switch (urb->actual_length) {
146
                      case 8:
147
                      case 10:
148
                      case 11:
149
                          // HID mouse input
                          hid_mouse_translate(kb->input.keys, &kb->
150
      input.rel_x, &kb->input.rel_y, -(urb->endpoint & 0xF), urb->actual_length, urb->buffer)
151
                          break;
152
                      case MSG_SIZE:
153
                          // Corsair mouse input
                          corsair_mousecopy(kb->input.keys, -(urb->endpoint & 0xF), urb
154
       ->buffer):
155
                          break;
156
157
                  } else if(IS_RGB(vendor, product)){
158
                      switch(urb->actual_length) {
159
                      case 8:
                          // RGB EP 1: 6KRO (BIOS mode) input
160
                          hid_kb_translate(kb->input.keys, -1, urb->actual_length, urb->
161
      buffer);
162
163
                      case 21:
164
                      case 5:
                          // RGB EP 2: NKRO (non-BIOS) input. Accept only if keyboard is inactive
165
                          if(!kb->active)
166
167
                               hid_kb_translate(kb->input.keys, -2, urb->actual_length,
      urb->buffer);
                      break;
case MSG_SIZE:
168
169
                          // RGB EP 3: Corsair input
170
                          corsair_kbcopy(kb->input.keys, -(urb->endpoint & 0xF), urb->
171
      buffer);
172
173
174
                 } else
                      // Non-RGB input
175
                      hid kb translate(kb->input.kevs, urb->endpoint & 0xF, urb->
176
      actual_length, urb->buffer);
177
                 inputupdate(kb);
178
                 pthread_mutex_unlock(imutex(kb));
179
                  // Re-submit the URB
                 ioctl(fd, USBDEVFS_SUBMITURB, urb);
180
181
                 urb = 0:
182
             }
183
184
         // Clean up
        for(int i = 0; i < urbcount; i++){
  ioctl(fd, USBDEVFS_DISCARDURB, urbs + i);
  free(urbs[i].buffer);</pre>
185
186
187
188
189
190
         return 0;
191 }
```



Here is the caller graph for this function:

## 8.42.2.8 int os\_resetusb ( usbdevice \* kb, const char \* file, int line )

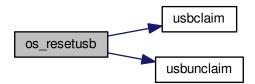
Definition at line 245 of file usb\_linux.c.

References usbdevice::handle, TEST\_RESET, usbclaim(), and usbunclaim().

Referenced by \_resetusb().

```
245
246 TEST_RESET(usbunclaim(kb, 1));
247 TEST_RESET(ioctl(kb->handle - 1, USBDEVFS_RESET));
248 TEST_RESET(usbclaim(kb));
249 // Success!
250 return 0;
251 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



8.42.2.9 void os\_sendindicators ( usbdevice \* kb )

Definition at line 86 of file usb\_linux.c.

References ckb\_err, usbdevice::handle, and usbdevice::ileds.

Referenced by updateindicators\_kb().

Here is the caller graph for this function:



8.42.2.10 int os\_setupusb ( usbdevice \* kb )

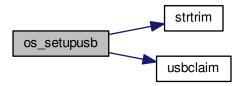
Definition at line 271 of file usb\_linux.c.

References ckb\_err, ckb\_info, ckb\_warn, devpath, usbdevice::epcount, FEAT\_RGB, usbdevice::fwversion, HAS\_-FEATURES, INDEX\_OF, KB\_NAME\_LEN, keyboard, usbdevice::name, usbdevice::serial, SERIAL\_LEN, strtrim(), usbdevice::udev, and usbclaim().

Referenced by \_setupusb().

```
271
272
        \ensuremath{//} Copy device description and serial
273
        struct udev_device* dev = kb->udev;
274
        const char* name = udev_device_get_sysattr_value(dev, "product");
275
        if (name)
276
            strncpy(kb->name, name, KB_NAME_LEN);
277
278
        const char* serial = udev_device_get_sysattr_value(dev, "serial");
279
        if(serial)
280
            strncpy(kb->serial, serial, SERIAL_LEN);
281
        strtrim(kb->serial);
        // Copy firmware version (needed to determine USB protocol)
```

```
283
        const char* firmware = udev_device_get_sysattr_value(dev, "bcdDevice");
284
        if(firmware)
285
             sscanf(firmware, "%hx", &kb->fwversion);
286
        else
        kb->fwversion = 0;
int index = INDEX_OF(kb, keyboard);
287
288
        ckb_info("Connecting %s at %s%d\n", kb->name, devpath, index);
289
290
291
        \ensuremath{//} Claim the USB interfaces
292
        const char* ep_str = udev_device_get_sysattr_value(dev, "bNumInterfaces");
293
        kb->epcount = 0;
294
        if(ep_str)
        sscanf(ep_str, "%d", &kb->epcount);
if(kb->epcount == 0){
295
296
297
            // This shouldn't happen, but if it does, assume EP count based on what the device is supposed to
            kb->epcount = (HAS_FEATURES(kb, FEAT_RGB) ? 4 : 3);
298
            ckb_warn("Unable to read endpoint count from udev, assuming d...\n", kb->
299
      epcount);
300
301
         if (usbclaim(kb)) {
302
             ckb\_err("Failed to claim interfaces: %s\n", strerror(errno));
303
             return -1;
304
305
        return 0;
306 }
```



Here is the caller graph for this function:



8.42.2.11 int os\_usbrecv ( usbdevice \* kb, uchar \* in\_msg, const char \* file, int line )

Definition at line 42 of file usb linux.c.

References ckb\_err\_fn, ckb\_warn\_fn, usbdevice::epcount, usbdevice::handle, and MSG\_SIZE.

Referenced by usbrecv().

```
42
43
       int res;
       // This is what CUE does, but it doesn't seem to work on linux.
44
       /*if(kb->fwversion >= 0x130){
45
           struct usbdevfs_bulktransfer transfer;
           memset(&transfer, 0, sizeof(transfer));
           transfer.ep = 0x84;
transfer.len = MSG_SIZE;
48
49
50
           transfer.timeout = 5000;
51
           transfer.data = in_msg;
           res = ioctl(kb->handle - 1, USBDEVFS_BULK, &transfer);
```

```
53
       } else {*/
            struct usbdevfs_ctrltransfer transfer = { 0xa1, 0x01, 0x0300, kb->
      epcount - 1, MSG_SIZE, 5000, in_msg };
    res = ioctl(kb->handle - 1, USBDEVFS_CONTROL, &transfer);
5.5
56
       if(res <= 0){
           ckb_err_fn("%s\n", file, line, res ? strerror(errno) : "No data read");
if(res == -1 && errno == ETIMEDOUT)
58
59
60
                 return -1;
61
62
                return 0;
       } else if(res != MSG_SIZE)
63
            ckb_warn_fn("Read %d bytes (expected %d)\n", file, line, res,
64
      MSG_SIZE);
65 #ifdef DEBUG_USB_RECV
     char converted[MSG_SIZE*3 + 1];
66
       for(int i=0;i<MSG_SIZE;i++)</pre>
67
            sprintf(&converted[i*3], "%02x ", in_msg[i]);
68
       ckb_warn_fn("Recv %s\n", file, line, converted);
69
70 #endif
       return res;
72 }
```

Here is the caller graph for this function:



8.42.2.12 int os\_usbsend ( usbdevice \* kb, const uchar \* out\_msg, int is\_recv, const char \* file, int line )

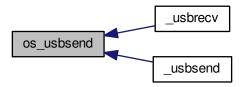
Definition at line 11 of file usb\_linux.c.

References ckb\_err\_fn, ckb\_warn\_fn, usbdevice::epcount, usbdevice::fwversion, usbdevice::handle, and MSG\_SI-ZE.

Referenced by usbrecv(), and usbsend().

```
11
12
        int res;
        if(kb->fwversion >= 0x120 && !is_recv){
    struct usbdevfs_bulktransfer transfer;
13
14
            memset(&transfer, 0, sizeof(transfer));
15
            transfer.ep = (kb \rightarrow fwversion >= 0x130 \&\& kb \rightarrow fwversion < 0x200) ? 4 : 3;
            transfer.len = MSG_SIZE;
17
18
            transfer.timeout = 5000;
19
            transfer.data = (void*)out_msq;
            res = ioctl(kb->handle - 1, USBDEVFS_BULK, &transfer);
20
       } else {
             struct usbdevfs_ctrltransfer transfer = { 0x21, 0x09, 0x0200, kb->
22
       epcount - 1, MSG_SIZE, 5000, (void*)out_msg };
    res = ioctl(kb->handle - 1, USBDEVFS_CONTROL, &transfer);
2.3
24
25
        if(res <= 0){
            ckb_err_fn("%s\n", file, line, res ? strerror(errno) : "No data written");
if(res == -1 && errno == ETIMEDOUT)
26
28
                 return -1;
29
30
                 return 0;
       } else if(res != MSG_SIZE)
31
32
             ckb_warn_fn("Wrote %d bytes (expected %d)\n", file, line, res,
       MSG_SIZE);
33 #ifdef DEBUG_USB
34
       char converted[MSG_SIZE*3 + 1];
        for(int i=0;i<MSG_SIZE;i++)</pre>
3.5
            sprintf(&converted[i*3], "%02x ", out_msq[i]);
36
37
        ckb_warn_fn("Sent %s\n", file, line, converted);
38 #endif
```

```
39    return res;
40 }
```



#### 8.42.2.13 const char\* product\_str ( short product )

Definition at line 26 of file usb.c.

References P\_K65, P\_K65\_LUX, P\_K65\_NRGB, P\_K65\_RFIRE, P\_K70, P\_K70\_LUX, P\_K70\_LUX\_NRGB, P\_K70\_NRGB, P\_K70\_NRGB, P\_K70\_RFIRE, P\_K70\_RFIRE\_NRGB, P\_K95, P\_K95\_NRGB, P\_K95\_PLATINUM, P\_M65, P\_M65\_PRO, P\_SABRE\_L, P\_SABRE\_N, P\_SABRE\_O, P\_SABRE\_O2, P\_SCIMITAR, P\_SCIMITAR\_PRO, P\_STRAFE, and P\_STRAFE\_NRGB.

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

```
if(product == P_K95 || product == P_K95_NRGB || product ==
       P_K95_PLATINUM)
       return "k95";
if(product == P_K70 || product == P_K70_NRGB || product ==
P_K70_LUX || product == P_K70_LUX_NRGB || product ==
28
2.9
       P_K70_RFIRE || product == P_K70_RFIRE_NRGB)
        if(product == P_K65 || product == P_K65_NRGB || product ==
31
       P_K65_LUX || product == P_K65_RFIRE)

return "k65";

if(product == P_STRAFE || product == P_STRAFE_NRGB)
32
33
             return "strafe";
34
        if(product == P_M65 || product == P_M65_PRO)
35
              return "m65";
        if(product == P_SABRE_0 || product == P_SABRE_L || product ==
37
       P_SABRE_N || product == P_SABRE_02)
    return "sabre";
38
        if(product == P_SCIMITAR || product == P_SCIMITAR_PRO)
    return "scimitar";
39
        return "";
41
42 }
```

Here is the caller graph for this function:



#### 8.42.2.14 int revertusb ( usbdevice \*kb )

Definition at line 137 of file usb.c.

References FEAT\_RGB, HAS\_FEATURES, NEEDS\_FW\_UPDATE, NK95\_HWON, nk95cmd, and setactive. Referenced by quitWithLock().

```
137
138
         if (NEEDS_FW_UPDATE(kb))
139
              return 0;
         if(!HAS_FEATURES(kb, FEAT_RGB)){
    nk95cmd(kb, NK95_HWON);
140
141
142
              return 0;
143
144
         if (setactive(kb, 0))
145
              return -1;
146
          return 0;
147 }
```

Here is the caller graph for this function:

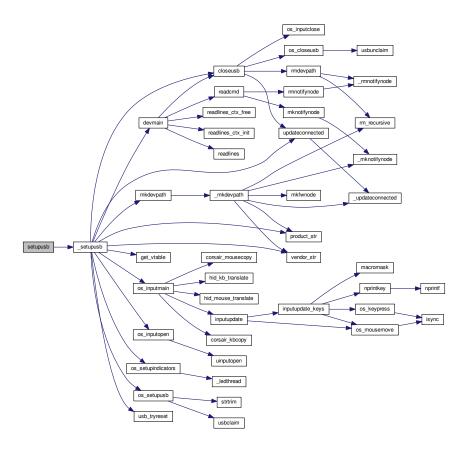


## 8.42.2.15 void setupusb ( usbdevice \* kb )

Definition at line 131 of file usb.c.

References \_setupusb(), ckb\_err, imutex, and usbdevice::thread.

Referenced by usbadd().



Here is the caller graph for this function:



```
8.42.2.16 int usb_tryreset ( usbdevice * kb )
```

Definition at line 164 of file usb.c.

References ckb\_err, ckb\_info, reset\_stop, and resetusb.

Referenced by \_setupusb(), and cmd\_fwupdate().

```
164
165
        if(reset_stop)
166
             return -1:
        ckb_info("Attempting reset...\n");
167
168
        while(1){
169
             int res = resetusb(kb);
170
             if(!res){
171
                ckb_info("Reset success\n");
172
                 return 0;
173
174
             <u>if</u>(res == -2 || reset_stop)
175
                 break;
176
```

```
177    ckb_err("Reset failed. Disconnecting.\n");
178    return -1;
179 }
```

Here is the caller graph for this function:

```
setupusb setupusb usbadd usb_add_device udev_enum udev_enum usb_main main restart cmd_restart
```

```
8.42.2.17 void usbkill ( )
```

Definition at line 492 of file usb linux.c.

Referenced by quitWithLock().

```
492 {
493 udev_unref(udev);
494 udev = 0;
495 }
```

Here is the caller graph for this function:



#### 8.42.2.18 int usbmain ( )

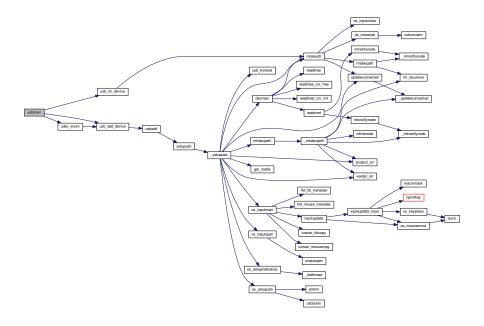
Definition at line 441 of file usb\_linux.c.

References ckb\_fatal, ckb\_warn, udev\_enum(), usb\_add\_device(), and usb\_rm\_device().

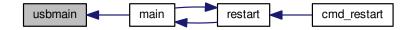
Referenced by main().

```
441
          // Load the uinput module (if it's not loaded already)
if(system("modprobe uinput") != 0)
    ckb_warn("Failed to load uinput module\n");
442
443
444
445
446
          // Create the udev object
447
          if(!(udev = udev_new())){
               ckb_fatal("Failed to initialize udev\n");
448
449
                return -1;
450
451
452
          // Enumerate all currently connected devices
453
454
          // Done scanning. Enter a loop to poll for device updates
struct udev_monitor* monitor = udev_monitor_new_from_netlink(udev, "udev");
455
456
457
          udev_monitor_filter_add_match_subsystem_devtype(monitor, "usb", 0);
458
          udev_monitor_enable_receiving(monitor);
459
          // Get an fd for the monitor
          int fd = udev_monitor_get_fd(monitor);
fd_set fds;
while (udev) {
460
461
462
463
               FD_ZERO(&fds);
464
               FD_SET(fd, &fds);
```

```
465
               // Block until an event is read
               if(select(fd + 1, &fds, 0, 0, 0) > 0 && FD_ISSET(fd, &fds)){
    struct udev_device* dev = udev_monitor_receive_device(monitor);
466
467
                    if(!dev)
468
469
                         continue;
470
                    const char* action = udev_device_get_action(dev);
471
                    if(!action){
472
                         udev_device_unref(dev);
473
                         continue;
474
475
                    // Add/remove device
                    if(!strcmp(action, "add")){
  int res = usb_add_device(dev);
  if(res == 0)
476
478
479
                               continue;
                         //\ \mbox{If the device matched but the handle wasn't opened correctly, re-enumerate (this
480
        sometimes solves the problem)
    if(res == -1)
    udev_enum();
481
482
483
                    } else if(!strcmp(action, "remove"))
484
                         usb_rm_device(dev);
485
                    udev_device_unref(dev);
486
487
488
         udev_monitor_unref(monitor);
489
         return 0;
490 }
```



Here is the caller graph for this function:



```
8.42.2.19 const char* vendor_str ( short vendor )
```

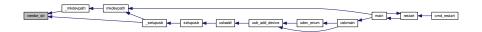
Definition at line 20 of file usb.c.

References V\_CORSAIR.

Referenced by mkdevpath(), and setupusb().

```
20
21    if(vendor == V_CORSAIR)
22        return "corsair";
23    return "";
24 }
```

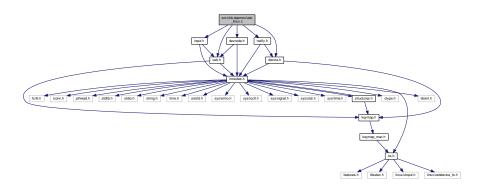
Here is the caller graph for this function:



## 8.43 src/ckb-daemon/usb\_linux.c File Reference

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

Include dependency graph for usb\_linux.c:



## **Data Structures**

• struct \_model

## **Macros**

- #define TEST\_RESET(op)
- #define N\_MODELS (sizeof(models) / sizeof(\_model))

#### **Functions**

int os\_usbsend (usbdevice \*kb, const uchar \*out\_msg, int is\_recv, const char \*file, int line)

- int os\_usbrecv (usbdevice \*kb, uchar \*in\_msg, const char \*file, int line)
- int \_nk95cmd (usbdevice \*kb, uchar bRequest, ushort wValue, const char \*file, int line)
- void os\_sendindicators (usbdevice \*kb)
- void \* os\_inputmain (void \*context)
- int usbunclaim (usbdevice \*kb, int resetting)
- void os\_closeusb (usbdevice \*kb)
- int usbclaim (usbdevice \*kb)
- int os\_resetusb (usbdevice \*kb, const char \*file, int line)
- void strtrim (char \*string)
- int os\_setupusb (usbdevice \*kb)
- int usbadd (struct udev device \*dev, short vendor, short product)
- static int usb\_add\_device (struct udev\_device \*dev)
- static void usb\_rm\_device (struct udev\_device \*dev)
- static void udev\_enum ()
- int usbmain ()
- · void usbkill ()

#### **Variables**

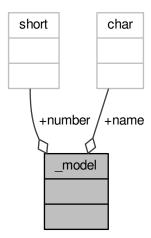
- static char kbsyspath [9][FILENAME\_MAX]
- static struct udev \* udev
- pthread\_t usbthread
- pthread\_t udevthread
- static \_model models []

#### 8.43.1 Data Structure Documentation

8.43.1.1 struct \_model

Definition at line 355 of file usb\_linux.c.

Collaboration diagram for \_model:



#### **Data Fields**

const char *	name	
short	number	

### 8.43.2 Macro Definition Documentation

## 8.43.2.1 #define N\_MODELS (sizeof(models) / sizeof(\_model))

Definition at line 386 of file usb linux.c.

Referenced by usb\_add\_device().

```
8.43.2.2 #define TEST_RESET( op )
```

#### Value:

Definition at line 237 of file usb\_linux.c.

Referenced by os\_resetusb().

#### 8.43.3 Function Documentation

8.43.3.1 int\_nk95cmd ( usbdevice \* kb, uchar bRequest, ushort wValue, const char \* file, int line )

Definition at line 74 of file usb\_linux.c.

References ckb\_err\_fn, usbdevice::handle, P\_K95\_NRGB, and usbdevice::product.

8.43.3.2 void os\_closeusb ( usbdevice \* kb )

Definition at line 214 of file usb\_linux.c.

References usbdevice::handle, INDEX\_OF, kbsyspath, keyboard, usbdevice::udev, and usbunclaim().

Referenced by closeusb().

```
214
        if(kb->handle){
215
            usbunclaim(kb, 0);
216
             close(kb->handle - 1);
218
219
        if (kb->udev)
220
            udev_device_unref(kb->udev);
        kb->handle = 0;
kb->udev = 0;
221
222
223
        kbsyspath[INDEX_OF(kb, keyboard)][0] = 0;
224 }
```



Here is the caller graph for this function:



#### 8.43.3.3 void\* os\_inputmain ( void \* context )

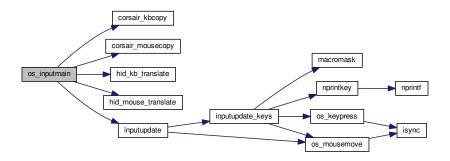
Definition at line 93 of file usb linux.c.

References usbdevice::active, ckb\_info, corsair\_kbcopy(), corsair\_mousecopy(), devpath, usbdevice::epcount, usbdevice::handle, hid\_kb\_translate(), hid\_mouse\_translate(), imutex, INDEX\_OF, usbdevice::input, inputupdate(), IS\_MOUSE, IS\_RGB, keyboard, usbinput::keys, MSG\_SIZE, usbdevice::product, usbinput::rel\_x, usbinput::rel\_y, and usbdevice::vendor.

Referenced by \_setupusb().

```
93
        usbdevice* kb = context;
94
95
        int fd = kb->handle - 1;
        short vendor = kb->vendor, product = kb->product;
96
        int index = INDEX_OF(kb, keyboard);
98
        ckb_info("Starting input thread for %s%d\n", devpath, index);
99
         // Monitor input transfers on all endpoints for non-RGB devices // For RGB, monitor all but the last, as it's used for input/output int urbcount = IS_RGB(vendor, product) ? (kb->epcount - 1) : kb->
101
102
       epcount;
103
         struct usbdevfs_urb urbs[urbcount];
104
         memset(urbs, 0, sizeof(urbs));
105
         urbs[0].buffer_length = 8;
         if(IS_RGB(vendor, product)) {
    if(IS_MOUSE(vendor, product))
106
107
108
                  urbs[1].buffer_length = 10;
109
110
                  urbs[1].buffer_length = 21;
              urbs[2].buffer_length = MSG_SIZE;
111
112
              if(urbcount != 3)
113
                   urbs[urbcount - 1].buffer length = MSG SIZE;
114
115
              urbs[1].buffer_length = 4;
116
              urbs[2].buffer_length = 15;
117
         // Submit URBs
118
         for(int i = 0; i < urbcount; i++) {</pre>
119
              urbs[i].type = USBDEVFS_URB_TYPE_INTERRUPT;
120
121
              urbs[i].endpoint = 0x80 | (i + 1);
122
              urbs[i].buffer = malloc(urbs[i].buffer_length);
123
              ioctl(fd, USBDEVFS_SUBMITURB, urbs + i);
124
125
         // Start monitoring input
126
         while(1){
127
              struct usbdevfs_urb* urb = 0;
```

```
128
             if(ioctl(fd, USBDEVFS_REAPURB, &urb)){
129
                 if(errno == ENODEV || errno == ENOENT || errno == ESHUTDOWN)
130
                      // Stop the thread if the handle closes
131
                      break;
                 else if(errno == EPIPE && urb){
    // On EPIPE, clear halt on the endpoint
132
133
134
                      ioctl(fd, USBDEVFS_CLEAR_HALT, &urb->endpoint);
135
                      // Re-submit the URB
136
                      if(urb)
                          ioctl(fd, USBDEVFS_SUBMITURB, urb);
137
                      urb = 0;
138
139
                 }
140
141
             if (urb) {
142
                  // Process input (if any)
143
                 pthread_mutex_lock(imutex(kb));
                  if(IS_MOUSE(vendor, product)) {
    switch(urb->actual_length) {
144
145
                      case 8:
146
147
                      case 10:
148
                      case 11:
149
                          // HID mouse input
                          hid_mouse_translate(kb->input.keys, &kb->
150
      input.rel\_x, \ \&kb->input.rel\_y, \ -(urb->endpoint \ \& \ 0xF), \ urb->actual\_length, \ urb->buffer)
151
                          break;
152
                      case MSG_SIZE:
153
                          // Corsair mouse input
154
                          corsair_mousecopy(kb->input.keys, -(urb->endpoint & 0xF), urb
      ->buffer);
155
                          break:
156
157
                  } else if(IS_RGB(vendor, product)){
158
                      switch(urb->actual_length) {
159
                      case 8:
                          // RGB EP 1: 6KRO (BIOS mode) input
160
                          hid_kb_translate(kb->input.keys, -1, urb->actual_length, urb->
161
      buffer);
162
                          break;
163
                      case 21:
164
                      case 5:
                          // RGB EP 2: NKRO (non-BIOS) input. Accept only if keyboard is inactive
165
                          if(!kb->active)
166
167
                               hid_kb_translate(kb->input.keys, -2, urb->actual_length,
      urb->buffer);
168
                          break;
169
                      case MSG_SIZE:
                          // RGB EP 3: Corsair input
170
171
                          corsair_kbcopy(kb->input.keys, -(urb->endpoint & 0xF), urb->
      buffer);
172
                          break;
173
174
                  } else
175
                      // Non-RGB input
                      hid_kb_translate(kb->input.keys, urb->endpoint & 0xF, urb->
176
      actual_length, urb->buffer);
177
                 inputupdate(kb);
178
                 pthread_mutex_unlock(imutex(kb));
179
                  // Re-submit the URB
                 ioctl(fd, USBDEVFS_SUBMITURB, urb);
180
                 urb = 0;
181
182
             }
183
184
         // Clean up
185
         ckb_info("Stopping input thread for s_d\n", devpath, index);
        for(int i = 0; i < urbcount; i++){
  ioctl(fd, USBDEVFS_DISCARDURB, urbs + i);
  free(urbs[i].buffer);</pre>
186
187
188
189
         return 0;
190
191 }
```



Here is the caller graph for this function:



## 8.43.3.4 int os\_resetusb ( usbdevice \* kb, const char \* file, int line )

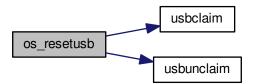
Definition at line 245 of file usb\_linux.c.

References usbdevice::handle, TEST\_RESET, usbclaim(), and usbunclaim().

Referenced by \_resetusb().

```
245
246 TEST_RESET(usbunclaim(kb, 1));
247 TEST_RESET(ioctl(kb->handle - 1, USBDEVFS_RESET));
248 TEST_RESET(usbclaim(kb));
249 // Success!
250 return 0;
```

Here is the call graph for this function:



Here is the caller graph for this function:



8.43.3.5 void os\_sendindicators ( usbdevice \* kb )

Definition at line 86 of file usb\_linux.c.

References ckb\_err, usbdevice::handle, and usbdevice::ileds.

Referenced by updateindicators\_kb().

Here is the caller graph for this function:



## 8.43.3.6 int os\_setupusb ( usbdevice \* kb )

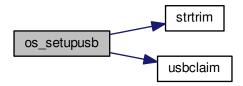
Definition at line 271 of file usb\_linux.c.

References ckb\_err, ckb\_info, ckb\_warn, devpath, usbdevice::epcount, FEAT\_RGB, usbdevice::fwversion, HAS\_-FEATURES, INDEX\_OF, KB\_NAME\_LEN, keyboard, usbdevice::name, usbdevice::serial, SERIAL\_LEN, strtrim(), usbdevice::udev, and usbclaim().

Referenced by \_setupusb().

```
271
272
        \ensuremath{//} Copy device description and serial
273
        struct udev_device* dev = kb->udev;
274
        const char* name = udev_device_get_sysattr_value(dev, "product");
275
        if (name)
276
            strncpy(kb->name, name, KB_NAME_LEN);
277
278
        const char* serial = udev_device_get_sysattr_value(dev, "serial");
279
        if(serial)
280
            strncpy(kb->serial, serial, SERIAL_LEN);
281
        strtrim(kb->serial);
        // Copy firmware version (needed to determine USB protocol)
```

```
283
        const char* firmware = udev_device_get_sysattr_value(dev, "bcdDevice");
284
        if(firmware)
285
             sscanf(firmware, "%hx", &kb->fwversion);
286
        kb->fwversion = 0;
int index = INDEX_OF(kb, keyboard);
287
288
        ckb_info("Connecting %s at %s%d\n", kb->name, devpath, index);
289
290
291
        \ensuremath{//} Claim the USB interfaces
292
        const char* ep_str = udev_device_get_sysattr_value(dev, "bNumInterfaces");
293
        kb->epcount = 0;
294
        if(ep_str)
        sscanf(ep_str, "%d", &kb->epcount);
if(kb->epcount == 0){
295
296
297
            // This shouldn't happen, but if it does, assume EP count based on what the device is supposed to
            kb->epcount = (HAS_FEATURES(kb, FEAT_RGB) ? 4 : 3);
298
            ckb_warn("Unable to read endpoint count from udev, assuming d...\n", kb->
299
      epcount);
300
301
         if (usbclaim(kb)) {
302
             ckb\_err("Failed to claim interfaces: %s\n", strerror(errno));
303
             return -1;
304
305
        return 0;
306 }
```



Here is the caller graph for this function:



8.43.3.7 int os\_usbrecv ( usbdevice \* kb, uchar \* in\_msg, const char \* file, int line )

Definition at line 42 of file usb linux.c.

References ckb\_err\_fn, ckb\_warn\_fn, usbdevice::epcount, usbdevice::handle, and MSG\_SIZE.

Referenced by usbrecv().

```
42
43
       int res;
       // This is what CUE does, but it doesn't seem to work on linux.
44
       /*if(kb->fwversion >= 0x130){
45
           struct usbdevfs_bulktransfer transfer;
           memset(&transfer, 0, sizeof(transfer));
           transfer.ep = 0x84;
transfer.len = MSG_SIZE;
48
49
50
           transfer.timeout = 5000;
51
           transfer.data = in_msg;
           res = ioctl(kb->handle - 1, USBDEVFS_BULK, &transfer);
```

```
53
       } else {*/
            struct usbdevfs_ctrltransfer transfer = { 0xa1, 0x01, 0x0300, kb->
      epcount - 1, MSG_SIZE, 5000, in_msg };
    res = ioctl(kb->handle - 1, USBDEVFS_CONTROL, &transfer);
5.5
56
       if(res <= 0){
           ckb_err_fn("%s\n", file, line, res ? strerror(errno) : "No data read");
if(res == -1 && errno == ETIMEDOUT)
58
59
60
                 return -1;
61
62
                return 0;
       } else if(res != MSG_SIZE)
63
            ckb_warn_fn("Read %d bytes (expected %d)\n", file, line, res,
64
      MSG_SIZE);
65 #ifdef DEBUG_USB_RECV
      char converted[MSG_SIZE*3 + 1];
66
       for(int i=0;i<MSG_SIZE;i++)</pre>
67
            sprintf(&converted[i*3], "%02x ", in_msg[i]);
68
       ckb_warn_fn("Recv %s\n", file, line, converted);
69
70 #endif
       return res;
72 }
```

Here is the caller graph for this function:



8.43.3.8 int os\_usbsend ( usbdevice \* kb, const uchar \* out\_msg, int is\_recv, const char \* file, int line )

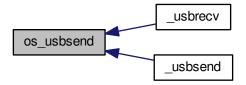
Definition at line 11 of file usb\_linux.c.

References ckb\_err\_fn, ckb\_warn\_fn, usbdevice::epcount, usbdevice::fwversion, usbdevice::handle, and MSG\_SI-ZE.

Referenced by usbrecv(), and usbsend().

```
11
12
        int res;
        if(kb->fwversion >= 0x120 && !is_recv){
    struct usbdevfs_bulktransfer transfer;
13
14
            memset(&transfer, 0, sizeof(transfer));
15
            transfer.ep = (kb \rightarrow fwversion >= 0x130 \&\& kb \rightarrow fwversion < 0x200) ? 4 : 3;
            transfer.len = MSG_SIZE;
17
18
            transfer.timeout = 5000;
19
            transfer.data = (void*)out_msq;
            res = ioctl(kb->handle - 1, USBDEVFS_BULK, &transfer);
20
       } else {
             struct usbdevfs_ctrltransfer transfer = { 0x21, 0x09, 0x0200, kb->
22
       epcount - 1, MSG_SIZE, 5000, (void*)out_msg };
    res = ioctl(kb->handle - 1, USBDEVFS_CONTROL, &transfer);
2.3
24
25
        if(res <= 0){
            ckb_err_fn("%s\n", file, line, res ? strerror(errno) : "No data written");
if(res == -1 && errno == ETIMEDOUT)
26
28
                 return -1;
29
30
                 return 0;
       } else if(res != MSG_SIZE)
31
32
             ckb_warn_fn("Wrote %d bytes (expected %d)\n", file, line, res,
       MSG_SIZE);
33 #ifdef DEBUG_USB
34
        char converted[MSG_SIZE*3 + 1];
        for(int i=0;i<MSG_SIZE;i++)</pre>
3.5
            sprintf(&converted[i*3], "%02x ", out_msq[i]);
36
37
        ckb_warn_fn("Sent %s\n", file, line, converted);
38 #endif
```

```
39     return res;
40 }
```



#### 8.43.3.9 void strtrim ( char \* string )

Definition at line 253 of file usb\_linux.c.

Referenced by os\_setupusb().

```
254
          // Find last non-space
         char* last = string;
for(char* c = string; *c != 0; c++){
255
256
2.57
              if(!isspace(*c))
258
                   last = c;
259
         last[1] = 0;
// Find first non-space
260
261
         char* first = string;
for(; *first != 0; first++) {
262
263
               if(!isspace(*first))
264
265
                   break:
266
267
          if (first != string)
268
              memmove(string, first, last - first);
269 }
```

Here is the caller graph for this function:



#### **8.43.3.10** static void udev\_enum ( ) [static]

Definition at line 418 of file usb\_linux.c.

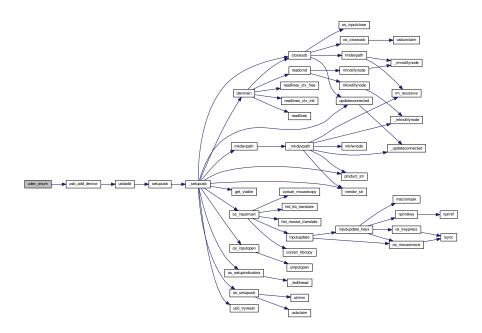
References usb\_add\_device(), and V\_CORSAIR\_STR.

Referenced by usbmain().

```
418 {
419 struct udev_enumerate* enumerator = udev_enumerate_new(udev);
420 udev_enumerate_add_match_subsystem(enumerator, "usb");
421 udev_enumerate_add_match_sysattr(enumerator, "idVendor", V_CORSAIR_STR);
422 udev_enumerate_scan_devices(enumerator);
423 struct udev_list_entry* devices, *dev_list_entry;
```

```
424
        devices = udev_enumerate_get_list_entry(enumerator);
425
426
        udev_list_entry_foreach(dev_list_entry, devices) {
427
             const char* path = udev_list_entry_get_name(dev_list_entry);
428
            if(!path)
429
                 continue:
430
            struct udev_device* dev = udev_device_new_from_syspath(udev, path);
431
            if(!dev)
432
                 continue;
             // If the device matches a recognized device ID, open it
433
            if(usb_add_device(dev))
    // Release device if not
434
435
436
                 udev_device_unref(dev);
437
438
        udev_enumerate_unref(enumerator);
439 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



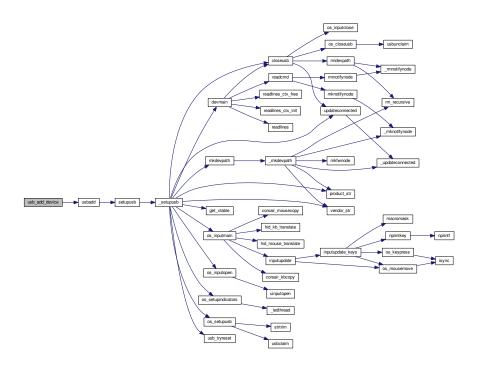
**8.43.3.11** static int usb\_add\_device ( struct udev\_device \* dev ) [static]

Definition at line 389 of file usb\_linux.c.

References N\_MODELS, usbadd(), V\_CORSAIR, and V\_CORSAIR\_STR.

Referenced by udev\_enum(), and usbmain().

```
392
              const char* product = udev_device_get_sysattr_value(dev, "idProduct");
              if(product){
393
394
                   for (_model* model = models; model < models +</pre>
       N_MODELS; model++) {
                       if(!strcmp(product, model->name)) {
    return usbadd(dev, V_CORSAIR, model->number);
395
396
397
398
399
400
401
         return 1;
402 }
```



Here is the caller graph for this function:



**8.43.3.12** static void usb\_rm\_device ( struct udev\_device \* dev ) [static]

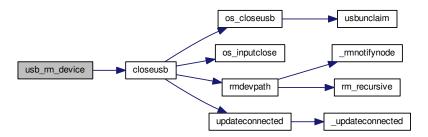
Definition at line 405 of file usb\_linux.c.

References closeusb(), DEV MAX, devmutex, kbsyspath, and keyboard.

Referenced by usbmain().

```
405
406  // Device removed. Look for it in our list of keyboards
407  const char* syspath = udev_device_get_syspath(dev);
408  if(!syspath || syspath[0] == 0)
409  return;
```

Here is the call graph for this function:



Here is the caller graph for this function:



## 8.43.3.13 int usbadd ( struct udev\_device \* dev, short vendor, short product )

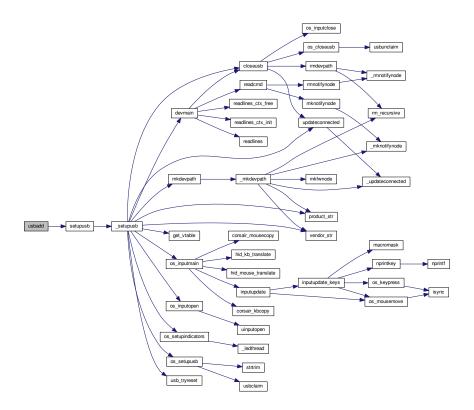
Definition at line 308 of file usb\_linux.c.

References ckb\_err, DEV\_MAX, dmutex, usbdevice::handle, IS\_CONNECTED, kbsyspath, keyboard, usbdevice::product, setupusb(), usbdevice::udev, and usbdevice::vendor.

Referenced by usb\_add\_device().

```
308
309
         const char* path = udev_device_get_devnode(dev);
         const char* syspath = udev_device_get_syspath(dev);
if(!path || !syspath || path[0] == 0 || syspath[0] == 0){
310
311
              ckb_err("Failed to get device path\n");
312
313
               return -1;
314
          // Find a free USB slot
315
         for(int index = 1; index < DEV_MAX; index++) {
    usbdevice* kb = keyboard + index;</pre>
316
317
               if(pthread_mutex_trylock(dmutex(kb))){
318
319
                    // If the mutex is locked then the device is obviously in use, so keep going
320
                    if(!strcmp(syspath, kbsyspath[index])){
                         // Make sure this existing keyboard doesn't have the same syspath (this shouldn't happen)
321
                         return 0:
322
323
324
                   continue;
325
326
               if(!IS_CONNECTED(kb)){
                   // Open the sysfs device
kb->handle = open(path, O_RDWR) + 1;
if(kb->handle <= 0){</pre>
327
328
329
330
                        ckb_err("Failed to open USB device: %s\n", strerror(errno));
331
                         kb->handle = 0;
```

```
332
                       pthread_mutex_unlock(dmutex(kb));
333
                       return -1;
334
                  } else {
                       // Set up device
kb->udev = dev;
335
336
                       kb->vendor = vendor;
kb->product = product;
337
338
339
                       strncpy(kbsyspath[index], syspath, FILENAME_MAX);
340
                       // Mutex remains locked
341
                       setupusb(kb);
342
                       return 0;
343
344
345
             pthread_mutex_unlock(dmutex(kb));
346
347
         ckb_err("No free devices\n");
348
         return -1;
349 }
```



Here is the caller graph for this function:



## 8.43.3.14 int usbclaim ( usbdevice \* kb )

Definition at line 226 of file usb\_linux.c.

References usbdevice::epcount, and usbdevice::handle.

Referenced by os\_resetusb(), and os\_setupusb().

```
226
227    int count = kb->epcount;
228    for(int i = 0; i < count; i++) {
229         struct usbdevfs_ioctl ctl = { i, USBDEVFS_DISCONNECT, 0 };
230         ioctl(kb->handle - 1, USBDEVFS_IOCTL, &ctl);
231         if(ioctl(kb->handle - 1, USBDEVFS_CLAIMINTERFACE, &i))
232         return -1;
233    }
234    return 0;
235 }
```

Here is the caller graph for this function:

```
8.43.3.15 void usbkill ( )
```

Definition at line 492 of file usb linux.c.

Referenced by quitWithLock().

```
492 {
493 udev_unref(udev);
494 udev = 0;
495 }
```

Here is the caller graph for this function:



```
8.43.3.16 int usbmain ( )
```

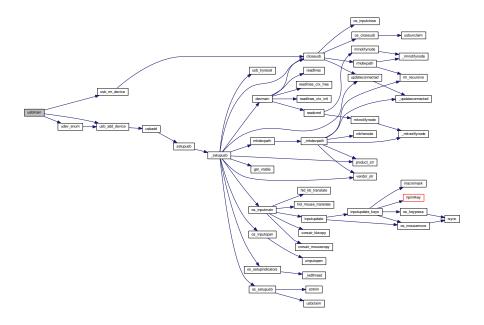
Definition at line 441 of file usb\_linux.c.

References ckb\_fatal, ckb\_warn, udev\_enum(), usb\_add\_device(), and usb\_rm\_device().

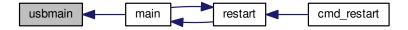
Referenced by main().

```
441
         // Load the uinput module (if it's not loaded already)
if(system("modprobe uinput") != 0)
442
443
444
             ckb_warn("Failed to load uinput module\n");
445
446
         // Create the udev object
         if(!(udev = udev_new())){
447
448
             ckb_fatal("Failed to initialize udev\n");
449
450
451
         \ensuremath{//} Enumerate all currently connected devices
452
453
         udev enum();
454
455
         // Done scanning. Enter a loop to poll for device updates
```

```
456
         struct udev_monitor* monitor = udev_monitor_new_from_netlink(udev, "udev");
457
         udev_monitor_filter_add_match_subsystem_devtype(monitor, "usb", 0);
458
        udev_monitor_enable_receiving(monitor);
459
         \ensuremath{//} Get an fd for the monitor
         int fd = udev_monitor_get_fd(monitor);
460
         fd_set fds;
461
462
        while (udev) {
463
             FD_ZERO(&fds);
464
             FD_SET(fd, &fds);
             // Block until an event is read
if(select(fd + 1, &fds, 0, 0, 0) > 0 && FD_ISSET(fd, &fds)){
    struct udev_device* dev = udev_monitor_receive_device(monitor);
465
466
467
468
                  if(!dev)
469
                       continue;
470
                  const char* action = udev_device_get_action(dev);
471
                  if(!action){
                       udev_device_unref(dev);
472
473
                       continue;
474
475
                  // Add/remove device
476
                  if(!strcmp(action, "add")){
                      int res = usb_add_device(dev);
if(res == 0)
477
478
                           continue;
479
480
                       // If the device matched but the handle wasn't opened correctly, re-enumerate (this
       sometimes solves the problem)
481
                       if(res == -1)
482
                           udev_enum();
                  } else if(!strcmp(action, "remove"))
483
484
                      usb_rm_device(dev);
485
                  udev_device_unref(dev);
486
487
488
         udev_monitor_unref(monitor);
489
         return 0;
490 }
```



Here is the caller graph for this function:



8.43.3.17 int usbunclaim ( usbdevice \* kb, int resetting )

Definition at line 193 of file usb linux.c.

References usbdevice::epcount, FEAT\_RGB, usbdevice::handle, and HAS\_FEATURES.

Referenced by os\_closeusb(), and os\_resetusb().

```
193
194
         int handle = kb->handle - 1;
        int count = kb->epcount;
for(int i = 0; i < count; i++)</pre>
195
196
197
             ioctl(handle, USBDEVFS_RELEASEINTERFACE, &i);
198
         // For RGB keyboards, the kernel driver should only be reconnected to interfaces 0 and 1 (HID), and
       only if we're not about to do a USB reset.
199
         \ensuremath{//} Reconnecting any of the others causes trouble.
200
        if(!resetting){
             struct usbdevfs_ioctl ctl = { 0, USBDEVFS_CONNECT, 0 };
201
             ioctl(handle, USBDEVFS_IOCTL, &ctl);
202
203
             ctl.ifno = 1;
204
             ioctl(handle, USBDEVFS_IOCTL, &ctl);
205
             // Also reconnect iface #2 (HID) for non-RGB keyboards
             if(!HAS_FEATURES(kb, FEAT_RGB)) {
   ctl.ifno = 2;
206
207
208
                  ioctl(handle, USBDEVFS_IOCTL, &ctl);
209
211
         return 0;
212 }
```

Here is the caller graph for this function:



## 8.43.4 Variable Documentation

**8.43.4.1 char kbsyspath[9][FILENAME\_MAX]** [static]

Definition at line 9 of file usb\_linux.c.

Referenced by os closeusb(), usb rm device(), and usbadd().

**8.43.4.2** \_model models[] [static]

Initial value:

```
{
    "lb17", 0x1b17 },
    { "lb07", 0x1b07 },
    { "lb37", 0x1b37 },
    { "lb39", 0x1b39 },
    { "lb13", 0x1b13 },
    { "lb38", 0x1b38 },
    { "lb38", 0x1b36 },
    { "lb38", 0x1b38 },
    { "lb38", 0x1b38 },
    { "lb11", 0x1b11 },
    { "lb2d", 0x1b2d },
    { "lb2d", 0x1b2d },
    { "lb15", 0x1b15 },

    { "lb12", 0x1b15 },

    { "lb12", 0x1b12 },
    { "lb15", 0x1b14 },
    { "lb16", 0x1b19 },
    { "lb19", 0x1b24 },
    { "lb19", 0x1b26 },
    { "lb19", 0x1b19 },
    { "lb19", 0x1b19 },
    { "lb19", 0x1b26 },
    { "lb19", 0x1b26 },
    { "lb19", 0x1b26 },
    { "lb19", 0x1b29 },
    { "lb19", 0x1b29 },
    { "lb19", 0x1b29 },
    { "lb19", 0x1b29 },
    { "lb19", 0x1b26 },
    { "lb19", 0x1b26 },
    { "lb32", 0x1b36 },
    { "lb32", 0x1b32 },
}
```

Definition at line 359 of file usb\_linux.c.

```
8.43.4.3 struct udev* udev [static]
```

Definition at line 351 of file usb\_linux.c.

#### 8.43.4.4 pthread\_t udevthread

Definition at line 352 of file usb\_linux.c.

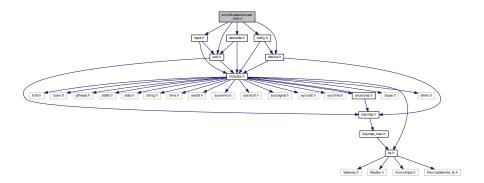
## 8.43.4.5 pthread\_t usbthread

Definition at line 352 of file usb\_linux.c.

## 8.44 src/ckb-daemon/usb\_mac.c File Reference

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

Include dependency graph for usb\_mac.c:



# Index

_model, 313	device.h, 47
_readlines_ctx, 65	ACT_M2
_DEFAULT_SOURCE	device.h, 47
os.h, <mark>220</mark>	ACT_M3
_GNU_SOURCE	device.h, 47
os.h, 220	ACT MR RING
FILE NOPATH	device.h, 47
includes.h, 104	ACT NEXT
_cmd_get	device.h, 47
notify.c, 207	ACT_NEXT_NOWRAP
_cmd_macro	device.h, 48
input.c, 107	active
•	
_freeprofile	devcmdunnamed, 26
profile.c, 222	allocprofile
_ledthread	devcmdunnamed, 26
input_linux.c, 125	profile.c, 222
_mkdevpath	profile.h, 238
devnode.c, 66	5.1.5
_mknotifynode	BIND
devnode.c, 68	command.h, 38
_nk95cmd	BR1
usb.h, 297	led_keyboard.c, 185
usb_linux.c, 314	BR2
_resetusb	led_keyboard.c, 185
usb.c, 277	BR4
usb.h, 297	led_keyboard.c, 185
_rmnotifynode	BTN WHEELDOWN
devnode.c, 69	keymap.h, 138
_setupusb	BTN_WHEELUP
usb.c, 277	keymap.h, 138
_start_dev	BUILD.md, 29
device.c, 44	BUTTON_HID_COUNT
updateconnected	keymap.c, 132
<b>–</b> ·	bind
devnode.c, 69	
_usbrecv	devcmdunnamed, 26
usb.c, 279	binding, 262
usb.h, 298	bit_reverse_table
_usbsend	led_keyboard.c, 192
usb.c, 280	CMD FIRST
usb.h, 299	CMD_FIRST
	command.h, 37
ACCEL	CMD_LAST
command.h, 37	command.h, 38
ACTIVE	CMD_VT_FIRST
command.h, 37	command.h, 37
ACT_LIGHT	CLEAR_KEYBIT
device.h, 47	structures.h, 271
ACT_LOCK	CMD_COUNT
device.h, 47	command.h, 36
ACT_M1	CMD_DEV_COUNT

command.h, 36	cmd_get
ckb_err	notify.c, 210
includes.h, 104	notify.h, 216
ckb_err_fn	cmd hwload kb
includes.h, 104	profile.h, 240
ckb_err_nofile	profile_keyboard.c, 252
includes.h, 104	cmd_hwload_mouse
ckb fatal	profile.h, 241
<del>-</del>	•
includes.h, 104	profile_mouse.c, 255
ckb_fatal_fn	cmd_hwsave_kb
includes.h, 104	profile.h, 242
ckb_fatal_nofile	profile_keyboard.c, 253
includes.h, 104	cmd_hwsave_mouse
ckb_info	profile.h, 243
includes.h, 104	profile_mouse.c, 256
ckb info fn	cmd iauto
includes.h, 104	led.c, 146
ckb info nofile	led.h, 151
includes.h, 105	cmd id
ckb_s_err	profile.c, 224
	profile.h, 244
includes.h, 105	•
ckb_s_out	cmd_idle_kb
includes.h, 105	device.h, 49
ckb_warn	device_keyboard.c, 56
includes.h, 105	cmd_idle_mouse
ckb_warn_fn	device.h, 50
includes.h, 105	device_mouse.c, 60
ckb_warn_nofile	cmd_inotify
includes.h, 105	led.c, 146
closeusb	led.h, 152
usb.c, 281	cmd_io_none
usb.h, 299	device vtable.c, 62
cmd	cmd ioff
command.h, 37	led.c, 146
cmd_active_kb	led.h, 152
device.h, 49	cmd_ion
device_keyboard.c, 55	led.c, 147
cmd_active_mouse	led.h, 153
device.h, 49	cmd_lift
device_mouse.c, 59	dpi.c, <mark>86</mark>
cmd_bind	dpi.h, 91
input.c, 109	cmd_macro
input.h, 117	input.c, 109
cmd_dpi	input.h, 117
_ · dpi.c, 85	cmd_macro_none
dpi.h, 90	device_vtable.c, 62
cmd_dpisel	cmd name
dpi.c, 86	profile.c, 225
•	•
dpi.h, 91	profile.h, 244
cmd_erase	cmd_none
profile.c, 223	device_vtable.c, 63
profile.h, 239	cmd_notify
cmd_eraseprofile	notify.c, 211
profile.c, 224	notify.h, 216
profile.h, 240	cmd_pollrate
and famindate	cina_poinate
cmd_fwupdate	device.h, 50
firmware.c, 96	
_ •	device.h, 50

profile.c, 225	NOTIFY, 38
profile.h, 245	NOTIFYOFF, 37
cmd_profilename	NOTIFYON, 37
profile.c, 226	POLLRATE, 37
profile.h, 245	PROFILEID, 37
cmd_rebind	PROFILENAME, 37
input.c, 110	REBIND, 38
input.h, 117	RESTART, 38
cmd_restart	RGB, 37 SCROLLSPEED, 37
notify.c, 211	SNAP, 38
notify.h, 217	SWITCH, 37
cmd_rgb led.c, 147	UNBIND, 38
led.h, 153	command.c
cmd snap	cmd_strings, 34
dpi.c, 86	readcmd, 30
dpi.h, 91	TRY_WITH_RESET, 30
cmd_strings	command.h
command.c, 34	CMD COUNT, 36
cmd unbind	CMD_DEV_COUNT, 36
input.c, 110	cmd, <mark>37</mark>
input.h, 118	cmdhandler, 36
cmdhandler	cmdhandler_io, 37
command.h, 36	cmdhandler_mac, 37
cmdhandler io	devcmd, 37
command.h, 37	readcmd, 39
cmdhandler mac	corsair_kbcopy
command.h, 37	keymap.c, 132
command.h	keymap.h, 141
ACCEL, 37	corsair_mousecopy
ACTIVE, 37	keymap.c, 133
BIND, 38	keymap.h, 141
CMD_FIRST, 37	DELAY
CMD_LAST, 38	command.h, 37
CMD_VT_FIRST, 37	DITHER
DELAY, 37	command.h, 37
DITHER, 37	DPI
DPI, 38	command.h, 38
DPISEL, 38	DPISEL
ERASE, 37	command.h, 38
ERASEPROFILE, 37	DAEMON.md, 29
FPS, 37	DELAY_LONG
FWUPDATE, 37	usb.h, 289
GET, 38	DELAY_MEDIUM
HWLOAD, 37	usb.h, 289
HWSAVE, 37	DELAY_SHORT
IAUTO, 38	usb.h, 289
ID, 37	DEV_MAX
IDLE, 37	device.h, 48
INOTIFY, 38	DPI_COUNT
IOFF, 37 ION, 37	structures.h, 271
LAYOUT, 37	devcmd, 35
LIFT, 38	command.h, 37 devcmdunnamed, 25
MACRO, 38	active, 26
MODE, 37	allocprofile, 26
NAME, 37	bind, 26
NONE, 37	dpi, 26
,	

dpisel, 26	IN_HID, 48
erase, 26	IS_CONNECTED, 48
eraseprofile, 26	imutex, 48
freeprofile, 26	inputmutex, 54
fwupdate, 26	keyboard, 55
get, <mark>26</mark>	setactive, 48
hwload, 26	setactive_kb, 50
hwsave, 26	setactive_mouse, 52
iauto, 27	setmodeindex_nrgb, 53
id, 27	start_dev, 54
idle, 27	start_kb_nrgb, 54
inotify, 27	device_keyboard.c
ioff, 27	cmd_active_kb, 55
ion, 27	cmd_idle_kb, 56
lift, 27	setactive_kb, 56
loadprofile, 27	setmodeindex_nrgb, 58
macro, 27	start kb nrgb, 58
name, 27	device mouse.c
notify, 27	cmd_active_mouse, 59
pollrate, 27	cmd_idle_mouse, 60
profileid, 27	cmd_pollrate, 60
profilename, 27	setactive_mouse, 60
rebind, 27	device_vtable.c
restart, 27	cmd_io_none, 62
rgb, 27	cmd_macro_none, 62
setmodeindex, 27	cmd_none, 63
snap, 27	int1_int_none, 63
start, 27	int1_void_none, 63
·	
unbind, 27	loadprofile_none, 63
updatedpi, 27	vtable_keyboard, 63
updateindicators, 27	vtable_keyboard_nonrgb, 63
updatergb, 27	vtable_mouse, 63
device.c	devlistmutex
_start_dev, 44	device.c, 45
devlistmutex, 45	devmain
devmutex, 45	usb.c, 282
hwload_mode, 45	devmutex
inputmutex, 45	device.c, 45
keyboard, 46	device.h, 54
start_dev, 45	devnode.c
device.h	_mkdevpath, 66
ACT_LIGHT, 47	_mknotifynode, 68
ACT_LOCK, 47	_rmnotifynode, 69
ACT_M1, 47	_updateconnected, 69
ACT_M2, 47	devpath, 76
ACT_M3, 47	gid, 76
ACT_MR_RING, 47	MAX_BUFFER, 65
ACT_NEXT, 47	mkdevpath, 70
ACT_NEXT_NOWRAP, 48	mkfwnode, 71
cmd_active_kb, 49	mknotifynode, 71
cmd_active_mouse, 49	readlines, 72
cmd_idle_kb, 49	readlines_ctx_free, 73
cmd_idle_mouse, 50	readlines_ctx_init, 73
cmd_pollrate, 50	rm_recursive, 73
DEV_MAX, 48	rmdevpath, 74
devmutex, 54	rmnotifynode, 75
dmutex, 48	S_GID_READ, 65
IN_CORSAIR, 48	updateconnected, 75

devnode.h	FPS
devpath, 84	command.h, 37
gid, 84	FWUPDATE
mkdevpath, 78	command.h, 37
mkfwnode, 79	FEAT_ADJRATE
mknotifynode, 80	structures.h, 271
readlines, 80	FEAT ANSI
readlines_ctx, 78	structures.h, 271
readlines_ctx_free, 81	FEAT BIND
readlines ctx init, 82	structures.h, 271
rmdevpath, 82	FEAT COMMON
rmnotifynode, 83	structures.h, 271
S CUSTOM, 78	FEAT FWUPDATE
S CUSTOM R, 78	structures.h, 271
S READ, 78	FEAT FWVERSION
S READDIR, 78	structures.h, 272
S READWRITE, 78	FEAT HWLOAD
updateconnected, 83	structures.h, 272
devpath	FEAT ISO
devnode.c, 76	structures.h, 272
devnode.h, 84	,
dmutex	FEAT_LMASK
device.h, 48	structures.h, 272
dpi	FEAT_MONOCHROME
devcmdunnamed, 26	structures.h, 272
dpi.c	FEAT_MOUSEACCEL
cmd_dpi, 85	structures.h, 272
cmd_dpisel, 86	FEAT_NOTIFY
cmd_lift, 86	structures.h, 272
cmd_snap, 86	FEAT_POLLRATE
loaddpi, 86	structures.h, 272
printdpi, 87	FEAT_RGB
savedpi, 88	structures.h, 272
•	FEAT_STD_NRGB
updatedpi, 89	structures.h, 272
dpi.h	FEAT_STD_RGB
cmd_dpi, 90	structures.h, 273
cmd_dpisel, 91	FW_MAXSIZE
cmd_lift, 91	firmware.c, 96
cmd_snap, 91	FW_NOFILE
loaddpi, 91	firmware.c, 96
printdpi, 92	FW OK
savedpi, 93	firmware.c, 96
updatedpi, 94	FW USBFAIL
dpisel	firmware.c, 96
devcmdunnamed, 26	FW WRONGDEV
dpiset, 263	firmware.c, 96
ERASE	features mask
command.h, 37	main.c, 206
ERASEPROFILE	usb.c, 286
command.h, 37	firmware.c
	cmd_fwupdate, 96
erase	FW MAXSIZE, 96
devcmdunnamed, 26	FW NOFILE, 96
eraseprofile	FW_NOFILE, 96 FW_OK, 96
devcmdunnamed, 26	FW_OK, 96 FW_USBFAIL, 96
euid_guard_start	<del>-</del>
os.h, 220	FW_WRONGDEV, 96
euid_guard_stop	fwupdate, 97
os.h, 220	getfwversion, 99

firmware.h	structures.h, 273
cmd_fwupdate, 100	HWMODE_OR_RETURN
getfwversion, 101	notify.c, 207
freebind	has_key
input.c, 110	led.c, 148
input.h, 118	hid_kb_translate
freemode	keymap.c, 133
profile.c, 226	keymap.h, 141
freeprofile	hid mouse translate
devcmdunnamed, 26	keymap.c, 135
profile.c, 227	keymap.h, 143
profile.h, 246	hwload
fwupdate	devcmdunnamed, 26
devcmdunnamed, 26	hwload_mode
firmware.c, 97	device.c, 45
11111Ware.c, 37	main.c, 206
GET	usb.c, 286
command.h, 38	hwloadmode
get devcmdunnamed, 26	profile_keyboard.c, 254
	hwloadprofile
get_vtable	profile.h, 238
usb.c, 283	hwprofile, 267
getfwversion	hwsave
firmware.c, 99	devcmdunnamed, 26
firmware.h, 101	hwtonative
gethwmodename	profile.c, 231
profile.c, 227	profile.h, 250
profile.h, 246	
gethwprofilename	IAUTO
profile.c, 228	command.h, 38
profile.c, 228 profile.h, 247	ID
profile.h, 247	
profile.h, 247 getid	ID
profile.h, 247 getid profile.c, 229	ID command.h, 37
profile.h, 247 getid profile.c, 229 profile.h, 248	ID command.h, 37 IDLE
profile.h, 247 getid profile.c, 229 profile.h, 248 getmodename	ID command.h, 37 IDLE command.h, 37
profile.h, 247 getid profile.c, 229 profile.h, 248 getmodename profile.c, 229	ID command.h, 37 IDLE command.h, 37 INOTIFY
profile.h, 247 getid     profile.c, 229     profile.h, 248 getmodename     profile.c, 229     profile.h, 248	ID command.h, 37 IDLE command.h, 37 INOTIFY command.h, 38 IOFF
profile.h, 247 getid     profile.c, 229     profile.h, 248 getmodename     profile.c, 229     profile.h, 248 getprofile.h, 248 getprofilename	ID command.h, 37 IDLE command.h, 37 INOTIFY command.h, 38 IOFF command.h, 37
profile.h, 247 getid     profile.c, 229     profile.h, 248 getmodename     profile.c, 229     profile.h, 248 getprofilename     profile.c, 230	ID command.h, 37 IDLE command.h, 37 INOTIFY command.h, 38 IOFF command.h, 37 ION
profile.h, 247 getid     profile.c, 229     profile.h, 248 getmodename     profile.c, 229     profile.h, 248 getprofilename     profile.c, 230     profile.h, 249	ID command.h, 37 IDLE command.h, 37 INOTIFY command.h, 38 IOFF command.h, 37 ION command.h, 37
profile.h, 247 getid     profile.c, 229     profile.h, 248 getmodename     profile.c, 229     profile.h, 248 getprofilename     profile.c, 230     profile.h, 249 gid	ID command.h, 37 IDLE command.h, 37 INOTIFY command.h, 38 IOFF command.h, 37 ION command.h, 37 I_CAPS
profile.h, 247 getid     profile.c, 229     profile.h, 248 getmodename     profile.c, 229     profile.h, 248 getprofilename     profile.c, 230     profile.h, 249 gid     devnode.c, 76	ID command.h, 37 IDLE command.h, 37 INOTIFY command.h, 38 IOFF command.h, 37 ION command.h, 37 I_CAPS structures.h, 273
profile.h, 247 getid     profile.c, 229     profile.h, 248 getmodename     profile.c, 229     profile.h, 248 getprofilename     profile.c, 230     profile.h, 249 gid	ID command.h, 37 IDLE command.h, 37 INOTIFY command.h, 38 IOFF command.h, 37 ION command.h, 37 I_CAPS structures.h, 273 I_NUM
profile.h, 247 getid     profile.c, 229     profile.h, 248 getmodename     profile.c, 229     profile.h, 248 getprofile.h, 248 getprofilename     profile.c, 230     profile.h, 249 gid     devnode.c, 76     devnode.h, 84	ID command.h, 37 IDLE command.h, 37 INOTIFY command.h, 38 IOFF command.h, 37 ION command.h, 37 I_CAPS structures.h, 273 I_NUM structures.h, 273
profile.h, 247 getid     profile.c, 229     profile.h, 248 getmodename     profile.c, 229     profile.h, 248 getprofile.h, 248 getprofilename     profile.c, 230     profile.h, 249 gid     devnode.c, 76     devnode.h, 84  HWLOAD	ID command.h, 37 IDLE command.h, 37 INOTIFY command.h, 38 IOFF command.h, 37 ION command.h, 37 I_CAPS structures.h, 273 I_NUM structures.h, 273 I_SCROLL
profile.h, 247 getid     profile.c, 229     profile.h, 248 getmodename     profile.c, 229     profile.h, 248 getprofile.h, 248 getprofilename     profile.c, 230     profile.h, 249 gid     devnode.c, 76     devnode.h, 84  HWLOAD     command.h, 37	ID command.h, 37 IDLE command.h, 37 INOTIFY command.h, 38 IOFF command.h, 37 ION command.h, 37 I_CAPS structures.h, 273 I_NUM structures.h, 273 I_SCROLL structures.h, 273
profile.h, 247 getid     profile.c, 229     profile.h, 248 getmodename     profile.c, 229     profile.h, 248 getprofilename     profile.c, 230     profile.h, 249 gid     devnode.c, 76     devnode.h, 84  HWLOAD     command.h, 37 HWSAVE	ID command.h, 37 IDLE command.h, 37 INOTIFY command.h, 38 IOFF command.h, 37 ION command.h, 37 I_CAPS structures.h, 273 I_NUM structures.h, 273 I_SCROLL structures.h, 273 IFACE_MAX
profile.h, 247 getid     profile.c, 229     profile.h, 248 getmodename     profile.c, 229     profile.h, 248 getprofilename     profile.c, 230     profile.h, 249 gid     devnode.c, 76     devnode.h, 84  HWLOAD     command.h, 37  HWSAVE     command.h, 37	ID command.h, 37 IDLE command.h, 37 INOTIFY command.h, 38 IOFF command.h, 37 ION command.h, 37 I_CAPS structures.h, 273 I_NUM structures.h, 273 I_SCROLL structures.h, 273 IFACE_MAX structures.h, 273
profile.h, 247 getid     profile.c, 229     profile.h, 248 getmodename     profile.c, 229     profile.h, 248 getprofilename     profile.c, 230     profile.h, 249 gid     devnode.c, 76     devnode.h, 84  HWLOAD     command.h, 37 HWSAVE     command.h, 37 HAS_ANY_FEATURE	ID command.h, 37 IDLE command.h, 37 INOTIFY command.h, 38 IOFF command.h, 37 ION command.h, 37 I_CAPS structures.h, 273 I_NUM structures.h, 273 I_SCROLL structures.h, 273 IFACE_MAX structures.h, 273 IN_CORSAIR
profile.h, 247 getid     profile.c, 229     profile.h, 248 getmodename     profile.c, 229     profile.h, 248 getprofilename     profile.c, 230     profile.h, 249 gid     devnode.c, 76     devnode.h, 84  HWLOAD     command.h, 37 HWSAVE     command.h, 37 HAS_ANY_FEATURE     structures.h, 273	ID command.h, 37 IDLE command.h, 37 INOTIFY command.h, 38 IOFF command.h, 37 ION command.h, 37 I_CAPS structures.h, 273 I_NUM structures.h, 273 I_SCROLL structures.h, 273 IFACE_MAX structures.h, 273 IN_CORSAIR device.h, 48
profile.h, 247 getid     profile.c, 229     profile.h, 248 getmodename     profile.c, 229     profile.h, 248 getprofile.h, 248 getprofilename     profile.c, 230     profile.h, 249 gid     devnode.c, 76     devnode.h, 84  HWLOAD     command.h, 37 HWSAVE     command.h, 37 HAS_ANY_FEATURE     structures.h, 273 HAS_FEATURES	ID command.h, 37 IDLE command.h, 37 INOTIFY command.h, 38 IOFF command.h, 37 ION command.h, 37 ION command.h, 37 I_CAPS structures.h, 273 I_NUM structures.h, 273 I_SCROLL structures.h, 273 IFACE_MAX structures.h, 273 IN_CORSAIR device.h, 48 IN_HID
profile.h, 247 getid     profile.c, 229     profile.h, 248 getmodename     profile.c, 229     profile.h, 248 getprofilename     profile.c, 230     profile.h, 249 gid     devnode.c, 76     devnode.h, 84  HWLOAD     command.h, 37 HWSAVE     command.h, 37 HAS_ANY_FEATURE     structures.h, 273 HAS_FEATURES     structures.h, 273	ID command.h, 37 IDLE command.h, 37 INOTIFY command.h, 38 IOFF command.h, 37 ION command.h, 37 I_CAPS structures.h, 273 I_NUM structures.h, 273 I_SCROLL structures.h, 273 IFACE_MAX structures.h, 273 IN_CORSAIR device.h, 48 IN_HID device.h, 48
profile.h, 247 getid     profile.c, 229     profile.h, 248 getmodename     profile.c, 229     profile.h, 248 getprofile.h, 248 getprofilename     profile.c, 230     profile.h, 249 gid     devnode.c, 76     devnode.h, 84  HWLOAD     command.h, 37 HWSAVE     command.h, 37 HAS_ANY_FEATURE     structures.h, 273 HAS_FEATURES	ID command.h, 37 IDLE command.h, 37 INOTIFY command.h, 38 IOFF command.h, 37 ION command.h, 37 ION command.h, 37 I_CAPS structures.h, 273 I_NUM structures.h, 273 I_SCROLL structures.h, 273 IFACE_MAX structures.h, 273 IN_CORSAIR device.h, 48 IN_HID
profile.h, 247 getid     profile.c, 229     profile.h, 248 getmodename     profile.c, 229     profile.h, 248 getprofilename     profile.c, 230     profile.h, 249 gid     devnode.c, 76     devnode.h, 84  HWLOAD     command.h, 37 HWSAVE     command.h, 37 HAS_ANY_FEATURE     structures.h, 273 HAS_FEATURES     structures.h, 273	ID command.h, 37 IDLE command.h, 37 INOTIFY command.h, 38 IOFF command.h, 37 ION command.h, 37 I_CAPS structures.h, 273 I_NUM structures.h, 273 I_SCROLL structures.h, 273 IFACE_MAX structures.h, 273 IN_CORSAIR device.h, 48 IN_HID device.h, 48
profile.h, 247 getid     profile.c, 229     profile.h, 248 getmodename     profile.c, 229     profile.h, 248 getprofilename     profile.c, 230     profile.h, 249 gid     devnode.c, 76     devnode.h, 84  HWLOAD     command.h, 37 HWSAVE     command.h, 37 HAS_ANY_FEATURE     structures.h, 273 HAS_FEATURES     structures.h, 273 HW_STANDARD	ID command.h, 37 IDLE command.h, 37 INOTIFY command.h, 38 IOFF command.h, 37 ION command.h, 37 I_CAPS structures.h, 273 I_NUM structures.h, 273 I_SCROLL structures.h, 273 IFACE_MAX structures.h, 273 IN_CORSAIR device.h, 48 IN_HID device.h, 48 INDEX_OF
profile.h, 247 getid     profile.c, 229     profile.h, 248 getmodename     profile.c, 229     profile.h, 248 getprofilename     profile.c, 230     profile.h, 249 gid     devnode.c, 76     devnode.h, 84  HWLOAD     command.h, 37 HWSAVE     command.h, 37 HAS_ANY_FEATURE     structures.h, 273 HAS_FEATURES     structures.h, 273 HW_STANDARD     notify.c, 207	ID command.h, 37 IDLE command.h, 37 INOTIFY command.h, 38 IOFF command.h, 37 ION command.h, 37 I_CAPS structures.h, 273 I_NUM structures.h, 273 I_SCROLL structures.h, 273 IFACE_MAX structures.h, 273 IN_CORSAIR device.h, 48 IN_HID device.h, 48 IN_HID includes.h, 105
profile.h, 247 getid     profile.c, 229     profile.h, 248 getmodename     profile.c, 229     profile.h, 248 getprofilename     profile.c, 230     profile.h, 249 gid     devnode.c, 76     devnode.h, 84  HWLOAD     command.h, 37 HWSAVE     command.h, 37 HAS_ANY_FEATURE     structures.h, 273 HAS_FEATURES     structures.h, 273 HW_STANDARD     notify.c, 207 HWMODE_K70     structures.h, 273	ID command.h, 37 IDLE command.h, 37 INOTIFY command.h, 38 IOFF command.h, 37 ION command.h, 37 I_CAPS structures.h, 273 I_NUM structures.h, 273 I_SCROLL structures.h, 273 IFACE_MAX structures.h, 273 IN_CORSAIR device.h, 48 IN_HID device.h, 48 INDEX_OF includes.h, 105 IS_CONNECTED device.h, 48
profile.h, 247 getid     profile.c, 229     profile.h, 248 getmodename     profile.c, 229     profile.h, 248 getprofilename     profile.c, 230     profile.h, 249 gid     devnode.c, 76     devnode.h, 84  HWLOAD     command.h, 37 HWSAVE     command.h, 37 HAS_ANY_FEATURE     structures.h, 273 HAS_FEATURES     structures.h, 273 HW_STANDARD     notify.c, 207 HWMODE_K70     structures.h, 273 HWMODE_K95	ID command.h, 37 IDLE command.h, 37 INOTIFY command.h, 38 IOFF command.h, 37 ION command.h, 37 I_CAPS structures.h, 273 I_NUM structures.h, 273 I_SCROLL structures.h, 273 IFACE_MAX structures.h, 273 IN_CORSAIR device.h, 48 IN_HID device.h, 48 INDEX_OF includes.h, 105 IS_CONNECTED device.h, 48 IS_FULLRANGE
profile.h, 247 getid     profile.c, 229     profile.h, 248 getmodename     profile.c, 229     profile.h, 248 getprofilename     profile.c, 230     profile.h, 249 gid     devnode.c, 76     devnode.h, 84  HWLOAD     command.h, 37 HWSAVE     command.h, 37 HAS_ANY_FEATURE     structures.h, 273 HAS_FEATURES     structures.h, 273 HW_STANDARD     notify.c, 207 HWMODE_K70     structures.h, 273	ID command.h, 37 IDLE command.h, 37 INOTIFY command.h, 38 IOFF command.h, 37 ION command.h, 37 I_CAPS structures.h, 273 I_NUM structures.h, 273 I_SCROLL structures.h, 273 IFACE_MAX structures.h, 273 IN_CORSAIR device.h, 48 IN_HID device.h, 48 INDEX_OF includes.h, 105 IS_CONNECTED device.h, 48

usb.h, 290	timespec_le, 106
IS_K70	timespec_lt, 106
usb.h, 290	uchar, 106
IS_K95	ushort, 106
usb.h, 290	initbind
IS_M65	input.c, 111
usb.h, 290	input.h, 118
IS_MOD	initmode
input.h, 116	profile.c, 231
IS_MONOCHROME	inotify
usb.h, 290	devcmdunnamed, 27
IS_MONOCHROME_DEV	input.c
usb.h, 290	_cmd_macro, 107
IS_MOUSE	cmd_bind, 109
usb.h, 290	cmd_macro, 109
IS_MOUSE_DEV	cmd_rebind, 110
usb.h, 291	cmd_unbind, 110
IS_RGB	freebind, 110
usb.h, 291	IS_WHEEL, 107
IS_RGB_DEV	initbind, 111
usb.h, 291	inputupdate, 111
IS_SABRE	inputupdate_keys, 112
usb.h, 291	macromask, 114
IS_SCIMITAR	updateindicators_kb, 115
usb.h, 291	input.h
IS_STRAFE	cmd_bind, 117
usb.h, 291	cmd_macro, 117
IS_WHEEL	cmd_rebind, 117
input.c, 107	cmd_unbind, 118
iauto	freebind, 118
devcmdunnamed, 27	IS_MOD, 116
id	initbind, 118
devcmdunnamed, 27	inputupdate, 119
idle	os_inputclose, 120
devcmdunnamed, 27	os_inputopen, 120
imutex	os_keypress, 121
device.h, 48	os_mousemove, 122
includes.h	os_setupindicators, 123
FILE_NOPATH, 104	updateindicators_kb, 123
ckb_err, 104	input_linux.c
ckb_err_fn, 104	_ledthread, 125
ckb_err_nofile, 104	isync, 126
ckb_fatal, 104	os_inputclose, 126
ckb_fatal_fn, 104	os_inputopen, 127
ckb_fatal_nofile, 104	os_keypress, 127
ckb_info, 104	os_mousemove, 128
ckb_info_fn, 104	os_setupindicators, 129
ckb_info_nofile, 105	uinputopen, 130
ckb_s_err, 105	inputmutex
ckb_s_out, 105	device.c, 45
ckb_warn, 105	device.h, 54
ckb_warn_fn, 105	inputupdate
ckb_warn_nofile, 105	input.c, 111
INDEX_OF, 105	input.h, 119
timespec_add, 106	inputupdate_keys
timespec_eq, 105	input.c, 112
timespec_ge, 105	int1_int_none
timespec_gt, 106	device_vtable.c, 63

int1_void_none	N_KEYBYTES_EXTENDED, 139
device_vtable.c, 63	N_KEYBYTES_HW, 139
ioff	N_KEYBYTES_INPUT, 140
devcmdunnamed, 27	N_KEYS_EXTENDED, 140
ion	N_KEYS_EXTRA, 140
devcmdunnamed, 27	N KEYS HW, 140
isblack	N KEYS INPUT, 140
led mouse.c, 194	N MOUSE ZONES, 140
iselect	SCAN KBD, 140
led.c, 148	SCAN MOUSE, 140
isync	SCAN SILENT, 140
input linux.c, 126	30/11/_5122.111, 110
	LAYOUT
KB_NAME_LEN	command.h, 37
structures.h, 274	LIFT
KEY BACKSLASH ISO	command.h, 38
keymap.h, 138	LED DPI
KEY CORSAIR	_ keymap.h, 139
keymap.h, 138	LED MOUSE
KEY NONE	keymap.h, 139
keymap.h, 138	LIFT MAX
KEY UNBOUND	<del>_</del>
<del>_</del>	structures.h, 274
keymap.h, 139	LIFT_MIN
kbsyspath	structures.h, 274
usb_linux.c, 328	led.c
key, 137	cmd_iauto, 146
keyboard	cmd_inotify, 146
device.c, 46	cmd_ioff, 146
device.h, 55	cmd_ion, 147
keymacro, 261	cmd_rgb, 147
keymap	has_key, 148
keymap.c, 136	iselect, 148
keymap.h, 144	printrgb, 149
keymap.c	led.h
BUTTON_HID_COUNT, 132	cmd_iauto, 151
corsair_kbcopy, 132	cmd_inotify, 152
corsair_mousecopy, 133	cmd_ioff, 152
hid_kb_translate, 133	cmd_ion, 153
hid_mouse_translate, 135	cmd_rgb, 153
keymap, 136	loadrgb_kb, 153
keymap.h	loadrgb_mouse, 155
BTN_WHEELDOWN, 138	printrgb, 156
BTN_WHEELUP, 138	savergb_kb, 157
corsair_kbcopy, 141	savergb_mouse, 159
corsair_mousecopy, 141	updatergb_kb, 159
hid_kb_translate, 141	updatergb_mouse, 161
hid_mouse_translate, 143	led_keyboard.c
KEY_BACKSLASH_ISO, 138	BR1, 185
KEY_CORSAIR, 138	BR2, 185
KEY_NONE, 138	BR4, 185
KEY_UNBOUND, 139	bit_reverse_table, 192
keymap, 144	loadrgb_kb, 186
LED_DPI, 139	makergb_512, 187
LED_MOUSE, 139	makergb_full, 188
MOUSE_BUTTON_FIRST, 139	O0, 185
MOUSE_EXTRA_FIRST, 139	O1, 185
N_BUTTONS_EXTENDED, 139	O2, 185
N_BUTTONS_HW, 139	O3, 185
N_KEY_ZONES, 139	O4, 185
_	

O5, 185	main
O6, 185	main.c, 198
O7, 185	main.c
O8, 185	features_mask, 206
ordered8to3, 189	hwload_mode, 206
quantize8to3, 189	localecase, 198
rgbcmp, 189	main, 198
savergb_kb, 190	main_ac, 206
updatergb_kb, 191	main_av, 206
led_mouse.c	quit, 201
isblack, 194	quitWithLock, 202
loadrgb_mouse, 195	reset_stop, 206
rgbcmp, 195	restart, 203
savergb_mouse, 196	sighandler, 204
updatergb_mouse, 196	sighandler2, 205
lift	timespec_add, 205
devcmdunnamed, 27	main_ac
lighting, 264	main.c, 206
loaddpi	main_av
dpi.c, 86	main.c, 206
dpi.h, 91 loadprofile	makergb_512 led keyboard.c, 187
devcmdunnamed, 27	makergb_full
profile.c, 232	led_keyboard.c, 188
profile.h, 250	mkdevpath
loadprofile_none	devnode.c, 70
device_vtable.c, 63	devnode.h, 78
loadrgb_kb	mkfwnode
led.h, 153	devnode.c, 71
led_keyboard.c, 186	devnode.h, 79
loadrgb_mouse	mknotifynode
led.h, 155	devnode.c, 71
led_mouse.c, 195	devnode.h, 80
localecase	models
main.c, 198	usb_linux.c, 328
MACRO	NAME
command.h, 38	command.h, 37
MODE	NONE
command.h, 37	command.h, 37
MACRO_MAX	NOTIFY
structures.h, 274	command.h, 38 NOTIFYOFF
MAX_BUFFER devnode.c, 65	command.h, 37
MD NAME LEN	NOTIFYON
structures.h, 274	command.h, 37
MODE COUNT	N_BUTTONS_EXTENDED
structures.h, 274	keymap.h, 139
MOUSE_BUTTON_FIRST	N BUTTONS HW
keymap.h, 139	keymap.h, 139
MOUSE EXTRA FIRST	N KEY ZONES
keymap.h, 139	keymap.h, 139
MSG_SIZE	N_KEYBYTES_EXTENDED
structures.h, 274	keymap.h, 139
macro	N_KEYBYTES_HW
devcmdunnamed, 27	keymap.h, 139
macroaction, 260	N_KEYBYTES_INPUT
macromask	keymap.h, 140
input.c, 114	N_KEYS_EXTENDED

1 440	*** 1 040
keymap.h, 140	notify.h, 219
N_KEYS_EXTRA keymap.h, 140	00
N KEYS HW	led_keyboard.c, 185
keymap.h, 140	O1
N KEYS INPUT	led_keyboard.c, 185
keymap.h, 140	O2
N_MODELS	led_keyboard.c, 185
usb_linux.c, 314	O3
N_MOUSE_ZONES	led_keyboard.c, 185
keymap.h, 140	04
NEEDS_FW_UPDATE	led_keyboard.c, 185
structures.h, 274	O5
NK95_HWOFF	led_keyboard.c, 185
usb.h, 291	O6 led_keyboard.c, 185
NK95_HWON	O7
usb.h, 291	led keyboard.c, 185
NK95_M1	O8
usb.h, 291	led_keyboard.c, 185
NK95_M2	OUTFIFO MAX
usb.h, 292	structures.h, 274
NK95_M3	ordered8to3
usb.h, 292	led keyboard.c, 189
name	os.h
devcmdunnamed, 27	_DEFAULT_SOURCE, 220
nativetohw	_GNU_SOURCE, 220
profile.c, 232	euid_guard_start, 220
profile.h, 251	euid_guard_stop, 220
nk95cmd	UINPUT_VERSION, 220
usb.h, 292	os_closeusb
notify	usb.h, 300
devcmdunnamed, 27	usb_linux.c, 314
notify.c	os_inputclose
_cmd_get, 207	input.h, 120
cmd_get, 210	input_linux.c, 126
cmd_notify, 211	os_inputmain
cmd_restart, 211	usb.h, 301
HW_STANDARD, 207	
HWMODE OR RETURN, 207	usb_linux.c, 315
	usb_linux.c, 315 os_inputopen
nprintf, 212	
nprintf, 212 nprintind, 212	os_inputopen
nprintf, 212 nprintind, 212 nprintkey, 213	os_inputopen input.h, 120
nprintf, 212 nprintind, 212 nprintkey, 213 restart, 214	os_inputopen input.h, 120 input_linux.c, 127
nprintf, 212 nprintind, 212 nprintkey, 213 restart, 214 notify.h	os_inputopen input.h, 120 input_linux.c, 127 os_keypress
nprintf, 212 nprintind, 212 nprintkey, 213 restart, 214 notify.h cmd_get, 216	os_inputopen input.h, 120 input_linux.c, 127 os_keypress input.h, 121
nprintf, 212 nprintind, 212 nprintkey, 213 restart, 214 notify.h cmd_get, 216 cmd_notify, 216	os_inputopen input.h, 120 input_linux.c, 127 os_keypress input.h, 121 input_linux.c, 127 os_mousemove input.h, 122
nprintf, 212 nprintind, 212 nprintkey, 213 restart, 214 notify.h cmd_get, 216 cmd_notify, 216 cmd_restart, 217	os_inputopen input.h, 120 input_linux.c, 127 os_keypress input.h, 121 input_linux.c, 127 os_mousemove input.h, 122 input_linux.c, 128
nprintf, 212 nprintind, 212 nprintkey, 213 restart, 214 notify.h cmd_get, 216 cmd_notify, 216 cmd_restart, 217 nprintf, 217	os_inputopen input.h, 120 input_linux.c, 127 os_keypress input.h, 121 input_linux.c, 127 os_mousemove input.h, 122
nprintf, 212 nprintind, 212 nprintkey, 213 restart, 214 notify.h cmd_get, 216 cmd_notify, 216 cmd_restart, 217 nprintf, 217 nprintind, 218	os_inputopen input.h, 120 input_linux.c, 127 os_keypress input.h, 121 input_linux.c, 127 os_mousemove input.h, 122 input_linux.c, 128 os_resetusb usb.h, 303
nprintf, 212 nprintind, 212 nprintkey, 213 restart, 214 notify.h cmd_get, 216 cmd_notify, 216 cmd_restart, 217 nprintf, 217 nprintind, 218 nprintkey, 219	os_inputopen input.h, 120 input_linux.c, 127 os_keypress input.h, 121 input_linux.c, 127 os_mousemove input.h, 122 input_linux.c, 128 os_resetusb usb.h, 303 usb_linux.c, 317
nprintf, 212 nprintind, 212 nprintkey, 213 restart, 214 notify.h cmd_get, 216 cmd_notify, 216 cmd_restart, 217 nprintf, 217 nprintind, 218 nprintkey, 219 nprintf	os_inputopen input.h, 120 input_linux.c, 127 os_keypress input.h, 121 input_linux.c, 127 os_mousemove input.h, 122 input_linux.c, 128 os_resetusb usb.h, 303 usb_linux.c, 317 os_sendindicators
nprintf, 212 nprintind, 212 nprintkey, 213 restart, 214 notify.h cmd_get, 216 cmd_notify, 216 cmd_restart, 217 nprintf, 217 nprintind, 218 nprintkey, 219 nprintf notify.c, 212	os_inputopen input.h, 120 input_linux.c, 127 os_keypress input.h, 121 input_linux.c, 127 os_mousemove input.h, 122 input_linux.c, 128 os_resetusb usb.h, 303 usb_linux.c, 317 os_sendindicators usb.h, 304
nprintf, 212 nprintind, 212 nprintkey, 213 restart, 214 notify.h cmd_get, 216 cmd_notify, 216 cmd_restart, 217 nprintf, 217 nprintind, 218 nprintkey, 219 nprintf notify.c, 212 notify.h, 217	os_inputopen input.h, 120 input_linux.c, 127 os_keypress input.h, 121 input_linux.c, 127 os_mousemove input.h, 122 input_linux.c, 128 os_resetusb usb.h, 303 usb_linux.c, 317 os_sendindicators usb.h, 304 usb_linux.c, 318
nprintf, 212 nprintind, 212 nprintkey, 213 restart, 214 notify.h cmd_get, 216 cmd_notify, 216 cmd_restart, 217 nprintf, 217 nprintind, 218 nprintkey, 219 nprintf notify.c, 212 notify.h, 217	os_inputopen input.h, 120 input_linux.c, 127 os_keypress input.h, 121 input_linux.c, 127 os_mousemove input.h, 122 input_linux.c, 128 os_resetusb usb.h, 303 usb_linux.c, 317 os_sendindicators usb.h, 304 usb_linux.c, 318 os_setupindicators
nprintf, 212 nprintind, 212 nprintkey, 213 restart, 214 notify.h cmd_get, 216 cmd_notify, 216 cmd_restart, 217 nprintf, 217 nprintind, 218 nprintkey, 219 nprintf notify.c, 212 notify.h, 217 nprintind notify.c, 212	os_inputopen input.h, 120 input_linux.c, 127 os_keypress input.h, 121 input_linux.c, 127 os_mousemove input.h, 122 input_linux.c, 128 os_resetusb usb.h, 303 usb_linux.c, 317 os_sendindicators usb.h, 304 usb_linux.c, 318 os_setupindicators input.h, 123
nprintf, 212 nprintind, 212 nprintkey, 213 restart, 214 notify.h cmd_get, 216 cmd_notify, 216 cmd_restart, 217 nprintf, 217 nprintind, 218 nprintkey, 219 nprintf notify.c, 212 notify.h, 217 nprintind notify.c, 212 notify.h, 218	os_inputopen input.h, 120 input_linux.c, 127 os_keypress input.h, 121 input_linux.c, 127 os_mousemove input.h, 122 input_linux.c, 128 os_resetusb usb.h, 303 usb_linux.c, 317 os_sendindicators usb.h, 304 usb_linux.c, 318 os_setupindicators input.h, 123 input_linux.c, 129
nprintf, 212 nprintind, 212 nprintkey, 213 restart, 214 notify.h cmd_get, 216 cmd_notify, 216 cmd_restart, 217 nprintf, 217 nprintind, 218 nprintkey, 219 nprintf notify.c, 212 notify.h, 217 nprintind notify.c, 212 notify.h, 218 nprintkey	os_inputopen input.h, 120 input_linux.c, 127 os_keypress input.h, 121 input_linux.c, 127 os_mousemove input.h, 122 input_linux.c, 128 os_resetusb usb.h, 303 usb_linux.c, 317 os_sendindicators usb.h, 304 usb_linux.c, 318 os_setupindicators input.h, 123 input_linux.c, 129 os_setupusb
nprintf, 212 nprintind, 212 nprintkey, 213 restart, 214 notify.h cmd_get, 216 cmd_notify, 216 cmd_restart, 217 nprintf, 217 nprintind, 218 nprintkey, 219 nprintf notify.c, 212 notify.h, 217 nprintind notify.c, 212 notify.h, 218	os_inputopen input.h, 120 input_linux.c, 127 os_keypress input.h, 121 input_linux.c, 127 os_mousemove input.h, 122 input_linux.c, 128 os_resetusb usb.h, 303 usb_linux.c, 317 os_sendindicators usb.h, 304 usb_linux.c, 318 os_setupindicators input.h, 123 input_linux.c, 129

usb. linux.c, 318  os_usbreev usb.h, 305 usb.h, 305 usb.h, 306 os_usbsend usb.h, 306 usb_linux.c, 320  POLIRATE command.h, 37 PROFILEID command.h, 37 PROFILEID command.h, 37 P_K65 usb.h, 292 P_K65_LUX usb.h, 292 P_K65_LUX usb.h, 292 P_K65_NRGB usb.h, 292 P_K65_NRGB STR usb.h, 292 P_K65_NRGB STR usb.h, 292 P_K65_NRGB usb.h, 292 P_K65_NRGB STR usb.h, 292 P_K65_NRGB Usb.h, 292 P_K65_NRGB STR usb.h, 292 P_K65_NRGB Usb.h, 293 P_K70_LUX usb.h, 293 P_K70_LUX_NRGB usb.h, 293 P_K70_NRGB_STR usb.h, 294 P_K70_RFIRE_NRGB_STR usb.h, 295 P_STRAFE_NRGB_STR usb.h, 296 P_STRAFE_NRGB_STR usb.h, 297 P_K70_RFIRE_NRGB_STR usb.h, 298 P_K70_RFIRE_STR usb.h, 299 P_K70_RFIRE_STR usb.h, 294 P_K70_RFIRE_STR usb.h, 295 P_STRAFE_STR usb.h, 296		
usb.h. 305 usb_linux.c, 319 Os_usbsend usb.h, 306 usb_linux.c, 320  P_ K95_PLATINUM_STR usb.h, 294 P_ K95_STR usb.h, 294 P_ K95_STR usb.h, 294 P_ K95_STR usb.h, 294 P_ M65_PRO usb_linux.c, 320  POLLRATE command.h, 37 PROFILEID command.h, 37 PROFILEID command.h, 37 PROFILENAME command.h, 37 P_ K65_PRO usb.h, 294 P_ K65_UX_STR usb.h, 292 P_ K65_LUX_STR usb.h, 292 P_ K65_LUX_STR usb.h, 292 P_ K65_NRGB usb.h, 292 P_ K65_RGB_STR usb.h, 292 P_ K65_RGB_STR usb.h, 292 P_ K65_RGB_STR usb.h, 292 P_ K65_RGB_STR usb.h, 292 P_ K65_RFIRE_STR usb.h, 292 P_ K65_RFIRE_STR usb.h, 292 P_ K65_STR usb.h, 293 P_ K70_LUX_NRGB_STR usb.h, 293 P_ K70_LUX_NRGB usb.h, 293 P_ K70_LUX_NRGB_STR usb.h, 293 P_ K70_LUX_NRGB_STR usb.h, 293 P_ K70_LUX_NRGB_STR usb.h, 293 P_ K70_LUX_STR usb.h, 293 P_ K70_RFIRE_NRGB usb.h, 293 P_ K70_RFIRE_NRGB usb.h, 294 P_ K70_RFIRE_NRGB usb.h, 296 P_ STRAFE_NRGB usb.h, 296 P_ STRAFE_NRGB usb.h, 296 P_ STRAFE_STR usb.h, 296 P_ K70_RFIRE_STR usb.h, 296 P_ STRAFE_STR usb.h, 296	<del>-</del>	•
usb.linux.c, 319         P_K95_PLATINUM_STR           0s_usbend         usb.h, 294           usb.l, 306         usb.h, 294           usb.l, 294         usb.h, 294           POLLRATE         usb.h, 294           command.h, 37         P_M65_PRO           PROFILEID         usb.h, 294           command.h, 37         P_M65_PRO_STR           p.K65         usb.h, 295           usb.h, 295         usb.h, 295           p.K65_LUX         usb.h, 295           p.K65_LUX STR         usb.h, 295           p.K65_NRGB         usb.h, 295           p.K65_NRGB STR         usb.h, 295           p.K65_RFIRE         usb.h, 295           p.K70_UX         p.SCIMITAR_PRO_U		
os_usbsend         usb.h, 294           usb.jinux.c, 320         P_K85_STR           usb.jinux.c, 320         usb.h, 294           POLLRATE         usb.h, 294           command.h, 37         P_M65_PRO           PROFILEID         usb.h, 294           command.h, 37         P_M65_PRO           PK65_PK65         usb.h, 295           P_K65_UX         P_M65_PRO           usb.h, 292         P_K65_LUX           usb.h, 292         P_SABRE_L           y-K65_NRGB         usb.h, 295           y-K65_NRGB         usb.h, 295           y-K65_NRGB         usb.h, 295           y-K65_NRGB STR         usb.h, 295           y-K65_NRGB STR         usb.h, 295           y-K65_RFIRE         p-SABRE_N STR           usb.h, 292         P_K65_RFIRE           y-K65_RFIRE         usb.h, 295           y-K65_RFIRE         p-SABRE_O_C STR           usb.h, 292         P_K65_RFIRE           y-K65_RFIRE         p-SABRE_O_C STR           usb.h, 293         P_SABRE_O_C STR           usb.h, 293         P_SABRE_O_C STR           usb.h, 293         P_SCIMITAR_PRO           y-K70_LUX_STR         p-SCIMITAR_PRO           usb.h, 293		
usb.h. 306 usb_linux.c, 320 usb_linux.c, 320  POLLRATE command.h, 37 PROFILEID command.h, 37 PROFILENAME command.h, 37 P_K65 usb.h, 294 command.h, 37 P_K65 usb.h, 295 P_K65_RFRE usb.h, 292 P_K65_LUX usb.h, 292 P_K65_LUX_STR usb.h, 292 P_K65_RRGB usb.h, 292 P_K65_RRGB usb.h, 292 P_K65_RRGB usb.h, 292 P_K65_RRGB usb.h, 292 P_K65_RFIRE usb.h, 293 P_K70_LUX_NRGB usb.h, 293 P_K70_LUX_STR usb.h, 293 P_K70_LUX_STR usb.h, 293 P_K70_LUX_NRGB usb.h, 293 P_K70_LUX_NRGB usb.h, 293 P_K70_LUX_NRGB usb.h, 293 P_K70_RFIRE_NRGB usb.h, 294 P_K70_RFIRE_NRGB_STR usb.h, 296 P_STRAFE_NRGB_STR usb.h, 296 P_STRAFE_NRGB_STR usb.h, 296 P_STRAFE_NRGB_Usb.h, 296 P_STRAFE_STR usb.h, 296 P_STRAFE_NGB_STR usb.h, 296 P_STRAFE_NGB_Unnamed, 27 P_K70_STR usb.h, 294 P_K70_STR usb.h, 294 P_K70_STR		P_K95_PLATINUM_STR
USD.İn, 294 POLLRATE command.h, 37 PROFILEID Command.h, 37 PROFILEID Command.h, 37 PROFILEID Command.h, 37 PROFILEID Command.h, 37 PROFILENAME Command.h, 39 PROFILENAME Command.h, 294 PROFILENAME Command.h, 295 PROFILENAME Command.h, 296 PROFILENAME Command.h,	os_usbsend	usb.h, 294
POLLRATE	usb.h, 306	P_K95_STR
POLITATE  command.h, 37  PROFILEID  command.h, 37  PROFILEID  command.h, 37  PROFILENAME  command.h, 37  P.K65  usb.h, 292  P.K65  usb.h, 292  P.K65_LUX  usb.h, 292  P.K65_LUX  usb.h, 292  P.K65_NRGB  usb.h, 292  P.K65_NRGB_STR  usb.h, 295  P.K65_STR  usb.h, 292  P.K65_STR  usb.h, 292  P.K65_STR  usb.h, 292  P.K65_STR  usb.h, 293  P.K70  usb.h, 293  P.K70  usb.h, 293  P.K70_LUX  usb.h, 293  P.K70_LUX, NRGB  usb.h, 293  P.K70_LUX, STR  usb.h, 296  P.SCIMITAR_PRO_STR  usb.h, 296  P.SCIMITAR_PRO_STR  usb.h, 296  P.STRAFE  usb.h, 296  P.STRAFE  usb.h, 296  P.STRAFE  usb.h, 296  P.STRAFE NRGB  usb.h, 296  P.STRAFE NRGB  usb.h, 296  P.STRAFE NRGB  usb.h, 296  P.STRAFE NRGB  usb.h, 296  P.STRAFE STR  u	usb_linux.c, 320	usb.h, 294
POLITATE  command.h, 37  PROFILEID  command.h, 37  PROFILEID  command.h, 37  PROFILENAME  command.h, 37  P.K65  usb.h, 292  P.K65  usb.h, 292  P.K65_LUX  usb.h, 292  P.K65_LUX  usb.h, 292  P.K65_NRGB  usb.h, 292  P.K65_NRGB_STR  usb.h, 295  P.K65_STR  usb.h, 292  P.K65_STR  usb.h, 292  P.K65_STR  usb.h, 292  P.K65_STR  usb.h, 293  P.K70  usb.h, 293  P.K70  usb.h, 293  P.K70_LUX  usb.h, 293  P.K70_LUX, NRGB  usb.h, 293  P.K70_LUX, STR  usb.h, 296  P.SCIMITAR_PRO_STR  usb.h, 296  P.SCIMITAR_PRO_STR  usb.h, 296  P.STRAFE  usb.h, 296  P.STRAFE  usb.h, 296  P.STRAFE  usb.h, 296  P.STRAFE NRGB  usb.h, 296  P.STRAFE NRGB  usb.h, 296  P.STRAFE NRGB  usb.h, 296  P.STRAFE NRGB  usb.h, 296  P.STRAFE STR  u		P M65
command.h, 37 PROFILEID command.h, 37 PROFILENAME command.h, 37 P_M65_PRO_STR usb.h, 295 P_K65 usb.h, 292 P_K65_LUX usb.h, 292 P_K65_LUX sub.h, 292 P_K65_LUX STR usb.h, 292 P_K65_NRGB usb.h, 292 P_K65_NRGB usb.h, 292 P_K65_NRGB_STR usb.h, 292 P_K65_RFIRE usb.h, 293 P_K65_STR usb.h, 293 P_K70 usb.h, 293 P_K70_LUX STR usb.h, 295 P_SCIMITAR PRO usb.h, 296 P_SCIMITAR_PRO_STR usb.h, 296 P_STRAFE usb.h, 296 P_STRAFE usb.h, 296 P_STRAFE NRGB usb.h, 296 P_STRAFE_NRGB usb.h, 296 P_STRAFE_NRGB_STR usb.h, 296 P_K70_RFIRE usb.h, 293 P_K70_RFIRE usb.h, 294 P_K70_RFIRE_NRGB_STR usb.h, 296 P_K70_RFIRE_NRGB_STR u	POLLRATE	<del>-</del>
PROFILEID  command h, 37  PROFILENAME command h, 37  P_K65		
command.h, 37 PROFILENAME command.h, 37 P,K65 usb.h, 292 P,K65 LUX usb.h, 292 P,K65_LUX usb.h, 292 P,K65_LUX usb.h, 292 P,K65_LUX str usb.h, 295 P,K65_LUX usb.h, 295 P,K65_LUX usb.h, 295 P,K65_LUX str usb.h, 295 P,K65_LUX Str usb.h, 295 P,K65_NRGB usb.h, 292 P,K65_NRGB usb.h, 292 P,K65_NRGB usb.h, 292 P,K65_RFIRE usb.h, 292 P,K65_RFIRE usb.h, 292 P,K65_RFIRE usb.h, 292 P,K65_RFIRE_STR usb.h, 293 P,K70 usb.h, 293 P,K70_LUX NRGB usb.h, 293 P,K70_LUX STR usb.h, 293 P,K70_LUX STR usb.h, 293 P,K70_LUX STR usb.h, 293 P,K70_LUX STR usb.h, 293 P,K70_NRGB P,STRAFE usb.h, 293 P,K70_NRGB STR usb.h, 293 P,K70_NRGB STR usb.h, 293 P,K70_NRGB STR usb.h, 293 P,K70_RFIRE NRGB usb.h, 296 P,STRAFE_NRGB usb.h, 296 P,K70_RFIRE_NRGB_STR us	PROFILEID	
PROFILENAME command.h, 37 P_K65 usb.h, 292 P_K65_LUX usb.h, 292 P_K65_LNSTR usb.h, 292 P_K65_NRGB usb.h, 292 P_K65_NRGB usb.h, 292 P_K65_NRGB_STR usb.h, 292 P_K65_RFIRE usb.h, 292 P_K65_RFIRE usb.h, 292 P_K65_RFIRE usb.h, 292 P_K65_RFIRE_STR usb.h, 293 P_K70_LUX usb.h, 293 P_K70_LUX_NRGB usb.h, 293 P_K70_LUX_STR usb.h, 293 P_K70_NRGB P_SCIMITAR_PRO_STR usb.h, 293 P_K70_NRGB P_STRAFE usb.h, 293 P_K70_NRGB P_STRAFE usb.h, 293 P_K70_RFIRE usb.h, 293 P_K70_RFIRE_NRGB usb.h, 294 P_K70_RFIRE_NRGB usb.h, 294 P_K70_RFIRE_STR usb.h, 294 P_K70_STR B usb.h, 294 P_K70_STR B usb.h, 294 P_K70_STR B usb.h, 294 P_K70_STR B usb.h, 294 P_K70_STR	command.h, 37	· ·
Command.h, 37 P_K65	PROFILENAME	
Usbh, 295	command.h, 37	
USDN, 292 P K85_LUX Usbh, 292 P K85_LUX STR Usbh, 292 P K85_NRGB Usbh, 292 P K85_RERE Usbh, 293 P K70 Usbh, 293 P K70 Usbh, 293 P K70_LUX Usbh, 293 P K70_LUX_NRGB Usbh, 293 P K70_LUX_NRGB Usbh, 293 P K70_LUX_NRGB Usbh, 293 P K70_LUX_STR Usbh, 293 P K70_LUX_STR Usbh, 293 P K70_LUX_STR Usbh, 293 P K70_NRGB Usbh, 293 P K70_RERE Usbh, 293 P K70_RERE Usbh, 293 P K70_RERE Usbh, 293 P K70_RERE Usbh, 294 P STRAFE_NRGB Usbh, 296 P STRAFE_STR Usbh	P_K65	
P. K85_LUX   usb.h, 295	usb.h, 292	
usb.h, 292 P_K65_LUX_STR usb.h, 292 P_K65_NRGB usb.h, 292 P_K65_NRGB_STR usb.h, 292 P_K65_NRGB_STR usb.h, 292 P_K65_RFIRE usb.h, 292 P_K65_RFIRE usb.h, 292 P_K65_STR usb.h, 292 P_K65_STR usb.h, 293 P_K70_LUX_NRGB usb.h, 293 P_K70_LUX_NRGB_STR usb.h, 293 P_K70_LUX_NRGB_STR usb.h, 293 P_K70_LUX_NRGB_STR usb.h, 293 P_K70_LUX_STR usb.h, 293 P_K70_NRGB_STR usb.h, 293 P_K70_NRGB usb.h, 293 P_K70_NRGB_STR usb.h, 293 P_K70_NRGB STR usb.h, 293 P_K70_NRGB_STR usb.h, 293 P_K70_NRGB_STR usb.h, 293 P_K70_NRGB_STR usb.h, 293 P_K70_NRGB_Usb.h, 296 P_STRAFE_NRGB usb.h, 296 P_STRAFE_NRGB usb.h, 296 P_STRAFE_NRGB usb.h, 296 P_STRAFE_NRGB usb.h, 296 P_STRAFE_STR usb.h, 296 P	P_K65_LUX	
P_K65_LUX_NRGB_STR  usb.h, 292 P_K66_RFIRE STR usb.h, 293 P_K70_LUX_NRGB P_SCIMITAR_PRO usb.h, 293 P_K70_LUX_NRGB P_SCIMITAR_PRO STR usb.h, 293 P_K70_LUX_NRGB P_SCIMITAR_PRO STR usb.h, 293 P_K70_LUX_NRGB P_SCIMITAR_PRO STR usb.h, 296 P_SCIMITAR_PRO_STR usb.h, 296 P_SCIMITAR_PRO_STR usb.h, 296 P_STRAFE_NRGB usb.h, 296 P_STRAFE usb.h, 296 P_STRAFE usb.h, 296 P_STRAFE_NRGB usb.h, 296 P_STRAFE_NRGB usb.h, 296 P_STRAFE_NRGB usb.h, 296 P_STRAFE_NRGB usb.h, 296 P_STRAFE_STR usb.h, 296 P_STRAFE_STR usb.h, 296 P_STRAFE_STR usb.h, 296 P_STRAFE_STR usb.h, 296 P_K70_RFIRE_STR usb.h, 294 P_K70_STR printame profile.c, 233 printrgb	usb.h, 292	
usb.h, 292 P_K65_NRGB	P K65 LUX STR	
P_K65_NRGB		
Usb.h, 292 P_K65_NRGB_STR Usb.h, 292 P_K65_RFIRE Usb.h, 292 P_K65_RFIRE Usb.h, 292 P_K65_RFIRE Usb.h, 292 P_K65_RFIRE_STR Usb.h, 292 P_K65_STR Usb.h, 293 P_K70_LUX_NRGB Usb.h, 293 P_K70_LUX_NRGB_STR Usb.h, 293 P_K70_LUX_STR Usb.h, 293 P_K70_NRGB_STR Usb.h, 293 P_K70_NRGB_STR Usb.h, 293 P_K70_RFIRE_Usb.h, 294 P_K70_RFIRE_Usb.h, 294 P_K70_RFIRE_Usb.h, 294 P_K70_STR Usb.h, 294 P_K70_STR Usb.h, 294 P_K95_NRGB Usb.h, 294 P_R70_RFIRE_NRGB Usb.h, 294 P_R70_RFIRE_NRGB Usb.h, 294 P_R70_STR P_R70_RFIRE_NRGB Usb.h, 294 P_R70_STR Usb.h, 294 P_R70_STR P_R70_RFIRE_DRGB P_R70_RFIRE_DR		
P_K65_NRGB_STR		
USD.h, 2992 P_K65_RFIRE USD.h, 292 P_K65_RFIRE_STR USD.h, 292 P_K65_RFIRE_STR USD.h, 293 P_K70 USD.h, 293 P_K70_LUX USD.h, 293 P_K70_LUX_NRGB USD.h, 293 P_K70_LUX_NRGB_STR USD.h, 293 P_K70_LUX_STR USD.h, 293 P_K70_NRGB P_SCIMITAR_PRO USD.h, 293 P_K70_RFIRE_NRGB STR USD.h, 293 P_K70_RFIRE_NRGB_STR USD.h, 294 P_K70_RFIRE_NRGB USD.h, 294 P_K95_NRGB USD.h, 294 Profile_c, 233 Printrgb	•	P_SABRE_N_STR
P_K65_RFIRE		usb.h, 295
usb.h, 292 P_K65_RFIRE_STR usb.h, 292 P_K65_STR usb.h, 293 P_K70 usb.h, 293 P_K70_LUX usb.h, 293 P_K70_LUX_NRGB usb.h, 293 P_K70_LUX_NRGB_STR usb.h, 293 P_K70_LUX_STR usb.h, 293 P_K70_NRGB P_K70_NRGB Usb.h, 293 P_K70_NRGB Usb.h, 293 P_K70_RFIRE_NRGB Usb.h, 293 P_K70_RFIRE_NRGB_STR Usb.h, 293 P_K70_RFIRE_NRGB Usb.h, 293 P_K70_RFIRE_NRGB Usb.h, 293 P_K70_RFIRE_NRGB Usb.h, 294 P_K70_RFIRE_NRGB Usb.h, 294 P_K70_STR Usb.h, 294 P_K95_NRGB		P_SABRE_O
P_K65_RFIRE_STR		usb.h, 295
usb.h, 292       usb.h, 295         P_K65_STR       usb.h, 295         usb.h, 293       P_SABRE_O_STR         p_K70       usb.h, 295         p_K70_LUX       usb.h, 295         usb.h, 293       P_SCIMITAR         p_K70_LUX_NRGB       usb.h, 295         usb.h, 293       P_SCIMITAR_PRO         usb.h, 293       p_SCIMITAR_PRO_STR         usb.h, 293       p_SCIMITAR_PRO_STR         usb.h, 293       p_SCIMITAR_STR         usb.h, 293       p_STRAFE         usb.h, 293       p_STRAFE         usb.h, 293       p_STRAFE_NRGB         usb.h, 293       p_STRAFE_NRGB         usb.h, 293       p_STRAFE_NRGB_STR         usb.h, 293       p_STRAFE_NRGB_STR         usb.h, 293       p_STRAFE_STR         usb.h, 293       p_STRAFE_STR         usb.h, 294       p_STRAFE_STR         usb.h, 294       pR_NAME_LEN         p_K70_RFIRE_STR       polirate         usb.h, 294       devcmdunnamed, 27         p_K95       dpi.h, 92         usb.h, 294       printname         p_K95       printname         usb.h, 294       printname         p_K95_NRGB       printname		P_SABRE_O2
P_K65_STR		usb.h, 295
Usb.h, 293		P SABRE O2 STR
P_K70		
usb.h, 293 P_K70_LUX usb.h, 293 P_K70_LUX_NRGB Usb.h, 293 P_K70_LUX_NRGB Usb.h, 293 P_K70_LUX_NRGB_STR Usb.h, 293 P_K70_LUX_NRGB_STR Usb.h, 293 P_K70_LUX_STR Usb.h, 293 P_K70_NRGB P_SCIMITAR_PRO_STR Usb.h, 293 P_K70_NRGB Usb.h, 293 P_K70_NRGB Usb.h, 293 P_K70_NRGB_STR Usb.h, 293 P_K70_RFIRE P_STRAFE_NRGB Usb.h, 293 P_K70_RFIRE P_STRAFE_NRGB Usb.h, 296 P_STRAFE_STR Usb.h, 296 P_STRAFE_STR Usb.h, 296 P_STRAFE_STR Usb.h, 296 P_K70_RFIRE_NRGB_STR Usb.h, 296 P_K70_RFIRE_STR Usb.h, 294 P_K70_STR Usb.h, 294 P_K70_STR Usb.h, 294 P_K95 Usb.h, 294 P_K95_NRGB Usb.h, 294 P_K95_NRGB Usb.h, 294 P_K95_NRGB Usb.h, 294 Printtgb		
P_K70_LUX	<del>-</del>	
usb.h, 293 P_K70_LUX_NRGB usb.h, 293 P_K70_LUX_NRGB usb.h, 293 P_K70_LUX_NRGB_STR usb.h, 293 P_K70_LUX_STR usb.h, 293 P_K70_NRGB usb.h, 293 P_K70_NRGB usb.h, 293 P_K70_NRGB_STR usb.h, 293 P_K70_NRGB_STR usb.h, 293 P_K70_NRGB_STR usb.h, 293 P_K70_RFIRE usb.h, 293 P_K70_RFIRE_NRGB usb.h, 293 P_K70_RFIRE_NRGB P_STRAFE_NRGB_STR usb.h, 293 P_K70_RFIRE_NRGB P_STRAFE_STR usb.h, 293 P_K70_RFIRE_NRGB Usb.h, 294 P_K70_RFIRE_STR usb.h, 294 P_K70_STR usb.h, 294 P_K85 usb.h, 294 P_K95_NRGB profile.c, 233 printrgb		
P_K70_LUX_NRGB		<del>-</del>
usb.h, 293       usb.h, 296         P_K70_LUX_NRGB_STR       usb.h, 296         usb.h, 293       usb.h, 296         P_K70_LUX_STR       usb.h, 296         usb.h, 293       usb.h, 296         P_K70_NRGB       usb.h, 296         usb.h, 293       usb.h, 296         P_K70_NRGB_STR       usb.h, 296         usb.h, 293       usb.h, 296         P_K70_RFIRE       P_STRAFE_NRGB         usb.h, 293       usb.h, 296         P_K70_RFIRE_NRGB       P_STRAFE_STR         usb.h, 293       usb.h, 296         P_K70_RFIRE_NRGB_STR       usb.h, 296         P_K70_RFIRE_NRGB_STR       usb.h, 296         P_K70_RFIRE_STR       pollrate         usb.h, 294       devcmdunnamed, 27         P_K70_STR       dpi.c, 87         usb.h, 294       printdpi         usb.h, 294       printname         P_K95       dpi.h, 92         usb.h, 294       printname         P_K95_NRGB       profile.c, 233         usb.h, 294       printrgb	,	
P_K70_LUX_NRGB_STR		
usb.h, 293       usb.h, 296         P_K70_LUX_STR       P_SCIMITAR_STR         usb.h, 293       usb.h, 296         P_K70_NRGB       usb.h, 296         usb.h, 293       usb.h, 296         P_K70_NRGB_STR       usb.h, 296         usb.h, 293       usb.h, 296         P_K70_RFIRE       P_STRAFE_NRGB_STR         usb.h, 293       usb.h, 296         P_K70_RFIRE_NRGB       P_STRAFE_STR         usb.h, 293       usb.h, 296         P_K70_RFIRE_NRGB_STR       usb.h, 296         P_K70_RFIRE_NRGB_STR       pR_NAME_LEN         usb.h, 294       structures.h, 275         P_K70_STR       pollrate         usb.h, 294       devcmdunnamed, 27         P_K95       dpi.h, 92         usb.h, 294       printname         P_K95_NRGB       profile.c, 233         usb.h, 294       profile.c, 233		
P_K70_LUX_STR		
usb.h, 293       usb.h, 296         P_K70_NRGB       P_STRAFE         usb.h, 293       usb.h, 296         P_K70_NRGB_STR       usb.h, 296         usb.h, 293       usb.h, 296         P_K70_RFIRE       P_STRAFE_NRGB_STR         usb.h, 293       usb.h, 296         P_K70_RFIRE_NRGB       P_STRAFE_STR         usb.h, 293       usb.h, 296         P_K70_RFIRE_NRGB_STR       usb.h, 296         P_K70_RFIRE_NRGB_STR       PR_NAME_LEN         usb.h, 294       structures.h, 275         P_K70_RFIRE_STR       pollrate         usb.h, 294       devcmdunnamed, 27         P_K70_STR       printdpi         usb.h, 294       printname         P_K95       dpi.h, 92         usb.h, 294       printname         P_K95_NRGB       profile.c, 233         usb.h, 294       printrgb		
P_K70_NRGB		
usb.h, 293       usb.h, 296         P_K70_NRGB_STR       P_STRAFE_NRGB         usb.h, 293       usb.h, 296         P_K70_RFIRE       P_STRAFE_NRGB_STR         usb.h, 293       usb.h, 296         P_K70_RFIRE_NRGB       P_STRAFE_STR         usb.h, 293       usb.h, 296         P_K70_RFIRE_NRGB_STR       usb.h, 296         PR_NAME_LEN       structures.h, 275         P_K70_RFIRE_STR       pollrate         usb.h, 294       devcmdunnamed, 27         P_K70_STR       printdpi         usb.h, 294       dpi.c, 87         P_K95       dpi.h, 92         usb.h, 294       printname         P_K95_NRGB       profile.c, 233         usb.h, 294       printrgb		
P_K70_NRGB_STR		<del>-</del>
usb.h, 293       usb.h, 296         P_K70_RFIRE       P_STRAFE_NRGB_STR         usb.h, 293       usb.h, 296         P_K70_RFIRE_NRGB       P_STRAFE_STR         usb.h, 293       usb.h, 296         P_K70_RFIRE_NRGB_STR       PR_NAME_LEN         usb.h, 294       structures.h, 275         P_K70_RFIRE_STR       pollrate         usb.h, 294       devcmdunnamed, 27         P_K70_STR       printdpi         usb.h, 294       dpi.c, 87         P_K95       dpi.h, 92         usb.h, 294       printname         P_K95_NRGB       profile.c, 233         usb.h, 294       printrgb	usb.h, 293	
P_K70_RFIRE	P_K70_NRGB_STR	
usb.h, 293       usb.h, 296         P_K70_RFIRE_NRGB       P_STRAFE_STR         usb.h, 293       usb.h, 296         P_K70_RFIRE_NRGB_STR       PR_NAME_LEN         usb.h, 294       structures.h, 275         P_K70_RFIRE_STR       pollrate         usb.h, 294       devcmdunnamed, 27         P_K70_STR       printdpi         usb.h, 294       dpi.c, 87         P_K95       dpi.h, 92         usb.h, 294       printname         P_K95_NRGB       profile.c, 233         usb.h, 294       printrgb	usb.h, 293	•
P_K70_RFIRE_NRGB       P_STRAFE_STR         usb.h, 293       usb.h, 296         P_K70_RFIRE_NRGB_STR       PR_NAME_LEN         usb.h, 294       structures.h, 275         P_K70_RFIRE_STR       pollrate         usb.h, 294       devcmdunnamed, 27         P_K70_STR       printdpi         usb.h, 294       dpi.c, 87         P_K95       dpi.h, 92         usb.h, 294       printname         P_K95_NRGB       profile.c, 233         usb.h, 294       printrgb	P_K70_RFIRE	
usb.h, 293       usb.h, 296         P_K70_RFIRE_NRGB_STR       PR_NAME_LEN         usb.h, 294       structures.h, 275         P_K70_RFIRE_STR       pollrate         usb.h, 294       devcmdunnamed, 27         P_K70_STR       printdpi         usb.h, 294       dpi.c, 87         P_K95       dpi.h, 92         usb.h, 294       printname         P_K95_NRGB       profile.c, 233         usb.h, 294       printrgb	usb.h, 293	
P_K70_RFIRE_NRGB_STR       PR_NAME_LEN         usb.h, 294       structures.h, 275         P_K70_RFIRE_STR       pollrate         usb.h, 294       devcmdunnamed, 27         P_K70_STR       printdpi         usb.h, 294       dpi.c, 87         P_K95       dpi.h, 92         usb.h, 294       printname         P_K95_NRGB       profile.c, 233         usb.h, 294       printrgb	P_K70_RFIRE_NRGB	P_STRAFE_STR
usb.h, 294       structures.h, 275         P_K70_RFIRE_STR       pollrate         usb.h, 294       devcmdunnamed, 27         P_K70_STR       printdpi         usb.h, 294       dpi.c, 87         P_K95       dpi.h, 92         usb.h, 294       printname         P_K95_NRGB       profile.c, 233         usb.h, 294       printrgb	usb.h, 293	usb.h, <mark>296</mark>
P_K70_RFIRE_STR       pollrate         usb.h, 294       devcmdunnamed, 27         P_K70_STR       printdpi         usb.h, 294       dpi.c, 87         P_K95       dpi.h, 92         usb.h, 294       printname         P_K95_NRGB       profile.c, 233         usb.h, 294       printrgb	P_K70_RFIRE_NRGB_STR	PR_NAME_LEN
usb.h, 294       devcmdunnamed, 27         P_K70_STR       printdpi         usb.h, 294       dpi.c, 87         P_K95       dpi.h, 92         usb.h, 294       printname         P_K95_NRGB       profile.c, 233         usb.h, 294       printrgb	usb.h, 294	structures.h, 275
usb.h, 294       devcmdunnamed, 27         P_K70_STR       printdpi         usb.h, 294       dpi.c, 87         P_K95       dpi.h, 92         usb.h, 294       printname         P_K95_NRGB       profile.c, 233         usb.h, 294       printrgb	P K70 RFIRE STR	pollrate
P_K70_STR       printdpi         usb.h, 294       dpi.c, 87         P_K95       dpi.h, 92         usb.h, 294       printname         P_K95_NRGB       profile.c, 233         usb.h, 294       printrgb		devcmdunnamed, 27
usb.h, 294       dpi.c, 87         P_K95       dpi.h, 92         usb.h, 294       printname         P_K95_NRGB       profile.c, 233         usb.h, 294       printrgb		
P_K95       dpi.h, 92         usb.h, 294       printname         P_K95_NRGB       profile.c, 233         usb.h, 294       printrgb		•
usb.h, 294 printname P_K95_NRGB profile.c, 233 usb.h, 294 printrgb		•
P_K95_NRGB profile.c, 233 usb.h, 294 printrgb	<del>-</del>	•
usb.h, 294 printrgb		•
		•
		. •

led.h, 156	hwloadmode, 254
product_str	profile_mouse.c
usb.c, 283	cmd_hwload_mouse, 255
usb.h, 307	cmd_hwsave_mouse, 256
profile.c	profileid
_freeprofile, 222	devcmdunnamed, 27
allocprofile, 222	profilename
cmd_erase, 223	devcmdunnamed, 27
cmd_eraseprofile, 224	
cmd_id, 224	quantize8to3
cmd_name, 225	led_keyboard.c, 189
cmd_profileid, 225	quit
cmd_profilename, 226	main.c, 201
freemode, 226	quitWithLock
freeprofile, 227	main.c, 202
gethwmodename, 227	REBIND
gethwprofilename, 228	
getid, 229	command.h, 38 RESTART
getmodename, 229	_
getprofilename, 230	command.h, 38 RGB
hwtonative, 231	command.h, 37
initmode, 231	README.md, 29
loadprofile, 232	
nativetohw, 232	ROADMAP.md, 29 readcmd
printname, 233	
setid, 234	command.c, 30 command.h, 39
u16dec, 234	readlines
u16enc, 235	devnode.c, 72
urldecode2, 235	devnode.h, 80
urlencode2, 236	readlines ctx
utf16to8, 237	devnode.h, 78
utf8to16, 237	readlines ctx free
profile.h	devnode.c, 73
allocprofile, 238	devnode.h, 81
cmd_erase, 239	readlines_ctx_init
cmd_eraseprofile, 240	devnode.c, 73
cmd_hwload_kb, 240	devnode.h, 82
cmd_hwload_mouse, 241	rebind
cmd_hwsave_kb, 242	devcmdunnamed, 27
cmd_hwsave_mouse, 243	reset_stop
cmd_id, 244	main.c, 206
cmd_name, 244	usb.c, 286
cmd profileid, 245	resetusb
cmd_profilename, 245	usb.h, 296
freeprofile, 246	restart
gethwmodename, 246	devcmdunnamed, 27
gethwprofilename, 247	main.c, 203
getid, 248	notify.c, 214
getmodename, 248	revertusb
getprofilename, 249	usb.c, 284
hwloadprofile, 238	usb.h, 307
hwtonative, 250	rgb
loadprofile, 250	devcmdunnamed, 27
nativetohw, 251	rgbcmp
setid, 251	led_keyboard.c, 189
profile_keyboard.c	led_mouse.c, 195
cmd_hwload_kb, 252	rm_recursive
cmd_hwsave_kb, 253	devnode.c, 73
·	•

rmdevpath	profile.c, 234
devnode.c, 74	profile.h, 251
devnode.h, 82	setmodeindex
rmnotifynode	devcmdunnamed, 27
devnode.c, 75	setmodeindex_nrgb
devnode.h, 83	device.h, 53
	device_keyboard.c, 58
SCROLLSPEED	setupusb
command.h, 37	usb.c, 284
SNAP	usb.h, 308
command.h, 38	sighandler
SWITCH	main.c, 204
command.h, 37	sighandler2
S_CUSTOM	_
devnode.h, 78	main.c, 205
S_CUSTOM_R	snap
devnode.h, 78	devcmdunnamed, 27
S_GID_READ	src/ckb-daemon/command.c, 29
devnode.c, 65	src/ckb-daemon/command.h, 34
S_READ	src/ckb-daemon/device.c, 43
devnode.h, 78	src/ckb-daemon/device.h, 46
S READDIR	src/ckb-daemon/device_keyboard.c, 55
devnode.h, 78	src/ckb-daemon/device_mouse.c, 59
S READWRITE	src/ckb-daemon/device_vtable.c, 62
<del>-</del>	src/ckb-daemon/devnode.c, 64
devnode.h, 78	src/ckb-daemon/devnode.h, 76
SCAN_KBD	src/ckb-daemon/dpi.c, 84
keymap.h, 140	src/ckb-daemon/dpi.h, 89
SCAN_MOUSE	src/ckb-daemon/extra_mac.c, 94
keymap.h, 140	src/ckb-daemon/firmware.c, 95
SCAN_SILENT	src/ckb-daemon/firmware.h, 100
keymap.h, 140	src/ckb-daemon/includes.h, 102
SCROLL_ACCELERATED	src/ckb-daemon/input.c, 106
structures.h, 275	src/ckb-daemon/input.h, 115
SCROLL_MAX	src/ckb-daemon/input linux.c, 124
structures.h, 275	src/ckb-daemon/input_mac.c, 131
SCROLL_MIN	• —
structures.h, 275	src/ckb-daemon/input_mac_mouse.c, 131
SERIAL_LEN	src/ckb-daemon/keymap.c, 131
structures.h, 275	src/ckb-daemon/keymap.h, 136
SET_KEYBIT	src/ckb-daemon/keymap_mac.h, 144
structures.h, 275	src/ckb-daemon/led.c, 145
savedpi	src/ckb-daemon/led.h, 151
dpi.c, 88	src/ckb-daemon/led_keyboard.c, 162
dpi.h, 93	src/ckb-daemon/led_mouse.c, 194
savergb_kb	src/ckb-daemon/main.c, 197
led.h, 157	src/ckb-daemon/notify.c, 206
led_keyboard.c, 190	src/ckb-daemon/notify.h, 215
savergb_mouse	src/ckb-daemon/os.h, 219
led.h, 159	src/ckb-daemon/profile.c, 221
led_mouse.c, 196	src/ckb-daemon/profile.h, 237
setactive	src/ckb-daemon/profile_keyboard.c, 252
device.h, 48	src/ckb-daemon/profile_mouse.c, 255
setactive_kb	src/ckb-daemon/structures.h, 257
device.h, 50	src/ckb-daemon/usb.c, 276
device_keyboard.c, 56	src/ckb-daemon/usb.h, 287
setactive_mouse	src/ckb-daemon/usb_linux.c, 312
device.h, 52	src/ckb-daemon/usb_mac.c, 329
device_mouse.c, 60	start
setid	devcmdunnamed, 27

start_dev	timespec_add
device.c, 45	includes.h, 106
device.h, 54	main.c, 205
start_kb_nrgb	timespec_eq
device.h, 54	includes.h, 105
device_keyboard.c, 58	timespec_ge
strtrim	includes.h, 105
usb_linux.c, 321	timespec_gt
structures.h	includes.h, 106
CLEAR KEYBIT, 271	timespec_le
DPI COUNT, 271	includes.h, 106
FEAT ADJRATE, 271	timespec_lt
FEAT_ANSI, 271	includes.h, 106
FEAT BIND, 271	,
FEAT COMMON, 271	u16dec
FEAT FWUPDATE, 271	profile.c, 234
FEAT FWVERSION, 272	u16enc
FEAT HWLOAD, 272	profile.c, 235
FEAT ISO, 272	UNBIND
FEAT LMASK, 272	command.h, 38
FEAT MONOCHROME, 272	UINPUT VERSION
FEAT MOUSEACCEL, 272	os.h, 220
FEAT NOTIFY, 272	USB_DELAY_DEFAULT
FEAT POLLRATE, 272	usb.h, 296
<del>-</del>	uchar
FEAT_RGB, 272	includes.h, 106
FEAT_STD_NRGB, 272	udev
FEAT_STD_RGB, 273	usb_linux.c, 329
HAS_ANY_FEATURE, 273	udev enum
HAS_FEATURES, 273	usb linux.c, 321
HWMODE_K70, 273	udevthread
HWMODE_K95, 273	usb_linux.c, 329
HWMODE_MAX, 273	uinputopen
I_CAPS, 273	input_linux.c, 130
I_NUM, 273	unbind
I_SCROLL, 273	
IFACE_MAX, 273	devcmdunnamed, 27
KB_NAME_LEN, 274	updateconnected
LIFT_MAX, 274	devnode.c, 75
LIFT_MIN, 274	devnode.h, 83
MACRO_MAX, 274	updatedpi
MD_NAME_LEN, 274	devcmdunnamed, 27
MODE_COUNT, 274	dpi.c, 89
MSG_SIZE, 274	dpi.h, 94
NEEDS_FW_UPDATE, 274	updateindicators
OUTFIFO_MAX, 274	devcmdunnamed, 27
PR_NAME_LEN, 275	updateindicators_kb
SCROLL_ACCELERATED, 275	input.c, 115
SCROLL_MAX, 275	input.h, 123
SCROLL_MIN, 275	updatergb
SERIAL_LEN, 275	devcmdunnamed, 27
SET_KEYBIT, 275	updatergb_kb
vtable_keyboard, 275	led.h, 159
vtable_keyboard_nonrgb, 275	led_keyboard.c, 191
vtable_mouse, 275	updatergb_mouse
	led.h, 161
TEST_RESET	led_mouse.c, 196
usb_linux.c, 314	urldecode2
TRY_WITH_RESET	profile.c, 235
command.c, 30	urlencode2

nunfile a COC	D KCE NDOD CTD 000
profile.c, 236	P_K65_NRGB_STR, 292
usb.c	P_K65_RFIRE, 292 P_K65_RFIRE_STR, 292
_resetusb, 277	
_setupusb, 277	P_K65_STR, 293
_usbrecv, 279 _usbsend, 280	P_K70, 293 P_K70_LUX, 293
closeusb, 281	P_K70_LUX, 293 P_K70_LUX_NRGB, 293
devmain, 282	P K70 LUX NRGB STR, 293
features_mask, 286	P K70 LUX STR, 293
get vtable, 283	P K70 NRGB, 293
hwload_mode, 286	P K70 NRGB STR, 293
product_str, 283	P K70 RFIRE, 293
reset_stop, 286	P_K70_RFIRE_NRGB, 293
revertusb, 284	P K70 RFIRE STR, 294
setupusb, 284	P K70 STR, 294
usb_tryreset, 285	P K95, 294
usbmutex, 287	P_K95_NRGB, 294
vendor_str, 286	P_K95_NRGB_STR, 294
usb.h	P K95 PLATINUM, 294
nk95cmd, 297	P_K95_PLATINUM_STR, 294
resetusb, 297	P K95 STR, 294
_usbrecv, 298	P M65, 294
_usbsend, 299	P_M65_PRO, 294
closeusb, 299	P_M65_PRO_STR, 295
DELAY_LONG, 289	P_M65_STR, 295
DELAY_MEDIUM, 289	P_SABRE_L, 295
DELAY_SHORT, 289	P SABRE L STR, 295
IS FULLRANGE, 290	P SABRE N, 295
IS_K65, 290	P_SABRE_N_STR, 295
IS_K70, 290	P_SABRE_O, 295
IS K95, 290	P SABRE 02, 295
IS_M65, 290	P_SABRE_O2_STR, 295
IS_MONOCHROME, 290	P_SABRE_O_STR, 295
IS_MONOCHROME_DEV, 290	P SCIMITAR, 295
IS_MOUSE, 290	P_SCIMITAR_PRO, 296
IS_MOUSE_DEV, 291	P_SCIMITAR_PRO_STR, 296
IS_RGB, 291	P_SCIMITAR_STR, 296
IS_RGB_DEV, 291	P_STRAFE, 296
IS_SABRE, 291	P_STRAFE_NRGB, 296
IS_SCIMITAR, 291	P_STRAFE_NRGB_STR, 296
IS_STRAFE, 291	P_STRAFE_STR, 296
NK95_HWOFF, 291	product_str, 307
NK95_HWON, 291	resetusb, 296
NK95_M1, 291	revertusb, 307
NK95_M2, 292	setupusb, 308
NK95_M3, 292	USB_DELAY_DEFAULT, 296
nk95cmd, 292	usb_tryreset, 309
os_closeusb, 300	usbkill, 310
os_inputmain, 301	usbmain, 310
os_resetusb, 303	usbrecv, 296
os_sendindicators, 304	usbsend, 297
os_setupusb, 304	V_CORSAIR, 297
os_usbrecv, 305	V_CORSAIR_STR, 297
os_usbsend, 306	vendor_str, 311
P_K65, 292	usb_add_device
P_K65_LUX, 292	usb_linux.c, 322
P_K65_LUX_STR, 292	usb_linux.c
P_K65_NRGB, 292	_nk95cmd, 314

kbsyspath, 328	profile.c, 237
models, 328	V COROAIR
N_MODELS, 314	V_CORSAIR
os_closeusb, 314	usb.h, 297
os_inputmain, 315	V_CORSAIR_STR
os_resetusb, 317	usb.h, 297
os_sendindicators, 318	vendor_str
os_setupusb, 318	usb.c, 286 usb.h, 311
os_usbrecv, 319	vtable_keyboard
os_usbsend, 320	device_vtable.c, 63
strtrim, 321	structures.h, 275
TEST_RESET, 314	vtable_keyboard_nonrgb
udev, 329	device_vtable.c, 63
udev_enum, 321	structures.h, 275
udevthread, 329	vtable mouse
usb_add_device, 322	device_vtable.c, 63
usb_rm_device, 323	structures.h, 275
usbadd, 324	5t. dotal 55, 275
usbclaim, 325	
usbkill, 326	
usbmain, 326	
usbthread, 329	
usbunclaim, 328	
usb_rm_device	
usb_linux.c, 323	
usb_tryreset	
usb.c, 285	
usb.h, 309	
usbadd	
usb_linux.c, 324	
usbclaim usb linux.c, 325	
usbdevice, 269	
usbid, 259	
usbinput, 268	
usbkill	
usb.h, 310	
usb_linux.c, 326	
usbmain	
usb.h, 310	
usb_linux.c, 326	
usbmode, 265	
usbmutex	
usb.c, 287	
usbprofile, 266	
usbrecv	
usb.h, 296	
usbsend	
usb.h, 297	
usbthread	
usb_linux.c, 329	
usbunclaim	
usb_linux.c, 328	
ushort	
includes.h, 106	
utf16to8	
profile.c, 237	
utf8to16	