ckb-next 0.2.7

Generated by Doxygen 1.8.6

Thu Apr 27 2017 21:14:34

Contents

1	Tod	o List			1				
2	Data	a Struct	ure Index		3				
	2.1	Data S	Structures		3				
3	File	Index			5				
	3.1	File Lis	File List						
4	Data	a Struct	ure Docun	nentation	7				
	4.1	devcm	dunnan	ned Struct Reference	7				
		4.1.1	Detailed	Description	8				
		4.1.2	Field Doo	cumentation	8				
			4.1.2.1	active	8				
			4.1.2.2	allocprofile	8				
			4.1.2.3	bind	8				
			4.1.2.4	dpi	8				
			4.1.2.5	dpisel	8				
			4.1.2.6	erase	8				
			4.1.2.7	eraseprofile	8				
			4.1.2.8	freeprofile	8				
			4.1.2.9	fwupdate	8				
			4.1.2.10	get	8				
			4.1.2.11	hwload	8				
			4.1.2.12	hwsave	9				
			4.1.2.13	iauto	9				
			4.1.2.14	id	9				
			4.1.2.15	idle	9				
			4.1.2.16	inotify	9				
			4.1.2.17	ioff	9				
				ion	9				
			4.1.2.19	lift	9				
			4 4 0 00		_				

iv CONTENTS

			4.1.2.21 ma	cro		 . 9
			4.1.2.22 na	me		 . 9
			4.1.2.23 no	ify		 . 9
			4.1.2.24 po	Irate		 . 9
			4.1.2.25 pro	fileid		 . 9
			4.1.2.26 pro	filename		 . 9
			4.1.2.27 rel	ind		 . 9
			4.1.2.28 res	tart		 . 9
			4.1.2.29 rgl			 . 9
			4.1.2.30 se	modeindex		 . 9
			4.1.2.31 sn	ap		 . 9
			4.1.2.32 sta	rt		 . 9
			4.1.2.33 un	oind		 . 9
			4.1.2.34 up	datedpi		 . 9
			4.1.2.35 up	dateindicators		 . 9
			4.1.2.36 up	datergb		 . 9
5	File	Docume	entation			11
•	5.1			mand.c File Reference		
	0.1	5.1.1		ion Documentation		
		•		Y_WITH_RESET		
		5.1.2		umentation		
				dcmd		
		5.1.3		umentation		
				d strings		
	5.2	src/ckb		mand.h File Reference		
		5.2.1		e Documentation		
				on devcmd		
		5.2.2		ion Documentation		
				ID_COUNT		
				ID DEV COUNT		
		5.2.3		mentation		
				dhandler		
				dhandler io		
				 dhandler_mac		
				/cmd		
		5.2.4		Type Documentation		
		-		d		
		5.2.5		umentation		
		-		dcmd		
					-	-

CONTENTS

5.3	src/ckb	-daemon/o	device.c File Reference	25
	5.3.1	Function	Documentation	26
		5.3.1.1	_start_dev	26
		5.3.1.2	start_dev	27
	5.3.2	Variable	Documentation	28
		5.3.2.1	devlistmutex	28
		5.3.2.2	devmutex	28
		5.3.2.3	hwload_mode	28
		5.3.2.4	inputmutex	28
		5.3.2.5	keyboard	28
5.4	src/ckb	-daemon/d	device.h File Reference	28
	5.4.1	Macro De	efinition Documentation	30
		5.4.1.1	ACT_LIGHT	30
		5.4.1.2	ACT_LOCK	30
		5.4.1.3	ACT_M1	30
		5.4.1.4	ACT_M2	30
		5.4.1.5	ACT_M3	30
		5.4.1.6	ACT_MR_RING	30
		5.4.1.7	ACT_NEXT	30
		5.4.1.8	ACT_NEXT_NOWRAP	30
		5.4.1.9	DEV_MAX	31
		5.4.1.10	dmutex	31
		5.4.1.11	imutex	31
		5.4.1.12	IN_CORSAIR	31
		5.4.1.13	IN_HID	31
		5.4.1.14	IS_CONNECTED	31
		5.4.1.15	setactive	31
	5.4.2	Function	Documentation	31
		5.4.2.1	cmd_active_kb	31
		5.4.2.2	cmd_active_mouse	32
		5.4.2.3	cmd_idle_kb	32
		5.4.2.4	cmd_idle_mouse	33
		5.4.2.5	cmd_pollrate	33
		5.4.2.6	setactive_kb	33
		5.4.2.7	setactive_mouse	35
		5.4.2.8	setmodeindex_nrgb	36
		5.4.2.9	start_dev	36
		5.4.2.10	start_kb_nrgb	37
	5.4.3	Variable	Documentation	37
		5.4.3.1	devmutex	37

vi CONTENTS

		5.4.3.2	inputmutex	37
		5.4.3.3	keyboard	37
5.5	src/ckb	o-daemon/	device_keyboard.c File Reference	38
	5.5.1	Function	Documentation	38
		5.5.1.1	cmd_active_kb	38
		5.5.1.2	cmd_idle_kb	39
		5.5.1.3	setactive_kb	39
		5.5.1.4	setmodeindex_nrgb	41
		5.5.1.5	start_kb_nrgb	41
5.6	src/ckb	o-daemon/	device_mouse.c File Reference	41
	5.6.1	Function	Documentation	42
		5.6.1.1	cmd_active_mouse	42
		5.6.1.2	cmd_idle_mouse	42
		5.6.1.3	cmd_pollrate	43
		5.6.1.4	setactive_mouse	43
5.7	src/ckb	o-daemon/	device_vtable.c File Reference	44
	5.7.1	Function	Documentation	45
		5.7.1.1	cmd_io_none	45
		5.7.1.2	cmd_macro_none	45
		5.7.1.3	cmd_none	46
		5.7.1.4	int1_int_none	46
		5.7.1.5	int1_void_none	46
		5.7.1.6	loadprofile_none	46
	5.7.2	Variable	Documentation	46
		5.7.2.1	vtable_keyboard	46
		5.7.2.2	vtable_keyboard_nonrgb	46
		5.7.2.3	vtable_mouse	46
5.8	src/ckb	o-daemon/	devnode.c File Reference	47
	5.8.1	Data Str	ucture Documentation	48
		5.8.1.1	struct _readlines_ctx	48
	5.8.2	Macro D	efinition Documentation	48
		5.8.2.1	MAX_BUFFER	48
		5.8.2.2	S_GID_READ	49
	5.8.3	Function	Documentation	49
		5.8.3.1	_mkdevpath	49
		5.8.3.2	_mknotifynode	51
		5.8.3.3	_rmnotifynode	52
		5.8.3.4	_updateconnected	52
		5.8.3.5	mkdevpath	53
		5.8.3.6	mkfwnode	54

CONTENTS vii

		5.8.3.7	mknotifynode	55
		5.8.3.8	readlines	55
		5.8.3.9	readlines_ctx_free	56
		5.8.3.10	readlines_ctx_init	57
		5.8.3.11	rm_recursive	57
		5.8.3.12	rmdevpath	58
		5.8.3.13	rmnotifynode	58
		5.8.3.14	updateconnected	59
	5.8.4	Variable I	Documentation	60
		5.8.4.1	devpath	60
		5.8.4.2	gid	60
5.9	src/ckb	-daemon/d	devnode.h File Reference	60
	5.9.1	Macro De	efinition Documentation	61
		5.9.1.1	S_CUSTOM	61
		5.9.1.2	S_CUSTOM_R	61
		5.9.1.3	S_READ	61
		5.9.1.4	S_READDIR	61
		5.9.1.5	S_READWRITE	61
	5.9.2	Typedef [Documentation	62
		5.9.2.1	readlines_ctx	62
	5.9.3	Function	Documentation	62
		5.9.3.1	mkdevpath	62
		5.9.3.2	mkfwnode	63
		5.9.3.3	mknotifynode	63
		5.9.3.4	readlines	64
		5.9.3.5	readlines_ctx_free	65
		5.9.3.6	readlines_ctx_init	65
		5.9.3.7	rmdevpath	66
		5.9.3.8	rmnotifynode	66
		5.9.3.9	updateconnected	67
	5.9.4	Variable I	Documentation	68
		5.9.4.1	devpath	68
		5.9.4.2	gid	68
5.10	src/ckb	-daemon/d	dpi.c File Reference	68
	5.10.1	Function	Documentation	68
		5.10.1.1	cmd_dpi	68
		5.10.1.2	cmd_dpisel	69
		5.10.1.3	$cmd_lift \ldots \ldots$	69
		5.10.1.4	cmd_snap	69
		5.10.1.5	loaddpi	70

viii CONTENTS

		5.10.1.6	printdpi	71
		5.10.1.7	savedpi	71
		5.10.1.8	updatedpi	72
5.11	src/ckb	-daemon/d	dpi.h File Reference	73
	5.11.1	Function	Documentation	73
		5.11.1.1	cmd_dpi	73
		5.11.1.2	cmd_dpisel	74
		5.11.1.3	cmd_lift	74
		5.11.1.4	cmd_snap	75
		5.11.1.5	loaddpi	75
		5.11.1.6	printdpi	76
		5.11.1.7	savedpi	76
		5.11.1.8	updatedpi	77
5.12	src/ckb	-daemon/e	extra_mac.c File Reference	78
5.13	src/ckb	-daemon/f	firmware.c File Reference	78
	5.13.1	Macro De	efinition Documentation	79
		5.13.1.1	FW_MAXSIZE	79
		5.13.1.2	FW_NOFILE	79
		5.13.1.3	FW_OK	79
		5.13.1.4	FW_USBFAIL	79
		5.13.1.5	FW_WRONGDEV	79
	5.13.2	Function	Documentation	79
		5.13.2.1	cmd_fwupdate	79
		5.13.2.2	fwupdate	80
		5.13.2.3	getfwversion	82
5.14	src/ckb	-daemon/f	firmware.h File Reference	83
	5.14.1	Function	Documentation	83
		5.14.1.1	cmd_fwupdate	84
		5.14.1.2	getfwversion	84
5.15	src/ckb	-daemon/i	includes.h File Reference	85
	5.15.1	Macro De	efinition Documentation	87
		5.15.1.1	FILE_NOPATH	87
		5.15.1.2	ckb_err	87
		5.15.1.3	ckb_err_fn	87
		5.15.1.4	ckb_err_nofile	87
		5.15.1.5	ckb_fatal	87
		5.15.1.6	ckb_fatal_fn	87
		5.15.1.7	ckb_fatal_nofile	87
		5.15.1.8	ckb_info	87
		5.15.1.9	ckb_info_fn	88

CONTENTS

	5.15.1.10 ckb_info_nofile	88
	5.15.1.11 ckb_s_err	88
	5.15.1.12 ckb_s_out	88
	5.15.1.13 ckb_warn	88
	5.15.1.14 ckb_warn_fn	88
	5.15.1.15 ckb_warn_nofile	88
	5.15.1.16 INDEX_OF	88
	5.15.1.17 timespec_eq	88
	5.15.1.18 timespec_ge	89
	5.15.1.19 timespec_gt	89
	5.15.1.20 timespec_le	89
	5.15.1.21 timespec_lt	89
5.15.	2 Typedef Documentation	89
	5.15.2.1 uchar	89
	5.15.2.2 ushort	89
5.15.	3 Function Documentation	89
	5.15.3.1 timespec_add	89
5.16 src/c	kb-daemon/input.c File Reference	89
5.16.	1 Macro Definition Documentation	90
	5.16.1.1 IS_WHEEL	90
5.16.	2 Function Documentation	90
	5.16.2.1 _cmd_macro	90
	5.16.2.2 cmd_bind	92
	5.16.2.3 cmd_macro	92
	5.16.2.4 cmd_rebind	93
	5.16.2.5 cmd_unbind	93
	5.16.2.6 freebind	93
	5.16.2.7 initbind	94
	5.16.2.8 inputupdate	94
	5.16.2.9 inputupdate_keys	95
	5.16.2.10 macromask	97
	5.16.2.11 updateindicators_kb	98
5.17 src/c	kb-daemon/input.h File Reference	98
5.17.	1 Macro Definition Documentation	99
	5.17.1.1 IS_MOD	99
5.17.	2 Function Documentation	100
	5.17.2.1 cmd_bind	100
	5.17.2.2 cmd_macro	100
	5.17.2.3 cmd_rebind	
	5.17.2.4 cmd_unbind	101

CONTENTS

		5.17.2.5	freebind	101
		5.17.2.6	initbind	101
		5.17.2.7	inputupdate	102
		5.17.2.8	os_inputclose	103
		5.17.2.9	os_inputopen	103
		5.17.2.10	os_keypress	104
		5.17.2.11	os_mousemove	105
		5.17.2.12	os_setupindicators	106
		5.17.2.13	B updateindicators_kb	107
5.18	src/ckb	-daemon/i	input_linux.c File Reference	107
	5.18.1	Function	Documentation	108
		5.18.1.1	_ledthread	108
		5.18.1.2	isync	109
		5.18.1.3	os_inputclose	109
		5.18.1.4	os_inputopen	110
		5.18.1.5	os_keypress	111
		5.18.1.6	os_mousemove	112
		5.18.1.7	os_setupindicators	112
		5.18.1.8	uinputopen	113
5.19	src/ckb	-daemon/i	nput_mac.c File Reference	114
			nput_mac_mouse.c File Reference	
5.21	src/ckb	-daemon/k	keymap.c File Reference	114
	5.21.1	Macro De	efinition Documentation	115
		5.21.1.1	BUTTON_HID_COUNT	115
	5.21.2	Function	Documentation	115
		5.21.2.1	corsair_kbcopy	115
		5.21.2.2	corsair_mousecopy	116
		5.21.2.3	hid_kb_translate	116
		5.21.2.4	hid_mouse_translate	118
	5.21.3	Variable I	Documentation	119
		5.21.3.1	keymap	119
5.22	src/ckb	-daemon/k	keymap.h File Reference	119
	5.22.1	Data Stru	ucture Documentation	120
		5.22.1.1	struct key	120
	5.22.2		efinition Documentation	
			BTN_WHEELDOWN	
			BTN_WHEELUP	
			KEY_BACKSLASH_ISO	
			KEY_CORSAIR	
		5.22.2.5	KEY_NONE	122

CONTENTS xi

		5.22.2.6 KEY_UNBOUND
		5.22.2.7 LED_DPI
		5.22.2.8 LED_MOUSE
		5.22.2.9 MOUSE_BUTTON_FIRST
		5.22.2.10 MOUSE_EXTRA_FIRST
		5.22.2.11 N_BUTTONS_EXTENDED
		5.22.2.12 N_BUTTONS_HW
		5.22.2.13 N_KEY_ZONES
		5.22.2.14 N_KEYBYTES_EXTENDED
		5.22.2.15 N_KEYBYTES_HW
		5.22.2.16 N_KEYBYTES_INPUT
		5.22.2.17 N_KEYS_EXTENDED
		5.22.2.18 N_KEYS_EXTRA
		5.22.2.19 N_KEYS_HW
		5.22.2.20 N_KEYS_INPUT
		5.22.2.21 N_MOUSE_ZONES
		5.22.2.22 N_MOUSE_ZONES_EXTENDED
		5.22.2.23 SCAN_KBD
		5.22.2.24 SCAN_MOUSE
		5.22.2.25 SCAN_SILENT
	5.22.3	Function Documentation
		5.22.3.1 corsair_kbcopy
		5.22.3.2 corsair_mousecopy
		5.22.3.3 hid_kb_translate
		5.22.3.4 hid_mouse_translate
	5.22.4	Variable Documentation
		5.22.4.1 keymap
5.23	src/ckb	-daemon/keymap_mac.h File Reference
5.24	src/ckb	-daemon/led.c File Reference
	5.24.1	Function Documentation
		5.24.1.1 cmd_iauto
		5.24.1.2 cmd_inotify
		5.24.1.3 cmd_ioff
		5.24.1.4 cmd_ion
		5.24.1.5 cmd_rgb
		5.24.1.6 has_key
		5.24.1.7 iselect
		5.24.1.8 printrgb
5.25	src/ckb	-daemon/led.h File Reference
	5.25.1	Function Documentation

xii CONTENTS

	5.25.1.1 cmd_iauto
	5.25.1.2 cmd_inotify
	5.25.1.3 cmd_ioff
	5.25.1.4 cmd_ion
	5.25.1.5 cmd_rgb
	5.25.1.6 loadrgb_kb
	5.25.1.7 loadrgb_mouse
	5.25.1.8 printrgb
	5.25.1.9 savergb_kb
	5.25.1.10 savergb_mouse
	5.25.1.11 updatergb_kb
	5.25.1.12 updatergb_mouse
	ckb-daemon/led_keyboard.c File Reference
5.26	.1 Macro Definition Documentation
	5.26.1.1 BR1
	5.26.1.2 BR2
	5.26.1.3 BR4
	5.26.1.4 O0
	5.26.1.5 O1
	5.26.1.6 O2
	5.26.1.7 O3
	5.26.1.8 O4
	5.26.1.9 O5
	5.26.1.10 O6
	5.26.1.11 O7
	5.26.1.12 O8
5.26	.2 Function Documentation
	5.26.2.1 loadrgb_kb
	5.26.2.2 makergb_512
	5.26.2.3 makergb_full
	5.26.2.4 ordered8to3
	5.26.2.5 quantize8to3
	5.26.2.6 rgbcmp
	5.26.2.7 savergb_kb
	5.26.2.8 updatergb_kb
5.26	.3 Variable Documentation
	5.26.3.1 bit_reverse_table
5.27 src/d	ckb-daemon/led_mouse.c File Reference
5.27	.1 Function Documentation
	5.27.1.1 isblack

CONTENTS xiii

		5.27.1.2	loadrgb_mouse	178
		5.27.1.3	rgbcmp	178
		5.27.1.4	savergb_mouse	179
		5.27.1.5	updatergb_mouse	179
5.28	src/ckb	-daemon/r	main.c File Reference	180
	5.28.1	Function	Documentation	181
		5.28.1.1	localecase	181
		5.28.1.2	main	182
		5.28.1.3	quit	184
		5.28.1.4	quitWithLock	185
		5.28.1.5	restart	186
		5.28.1.6	sighandler	187
		5.28.1.7	sighandler2	188
		5.28.1.8	timespec_add	188
	5.28.2	Variable I	Documentation	189
		5.28.2.1	features_mask	189
		5.28.2.2	hwload_mode	189
		5.28.2.3	main_ac	189
		5.28.2.4	main_av	189
		5.28.2.5	reset_stop	189
5.29			notify.c File Reference	
	5.29.1	Macro De	efinition Documentation	190
		5.29.1.1	HW_STANDARD	190
		5.29.1.2	HWMODE_OR_RETURN	190
	5.29.2	Function	Documentation	191
		5.29.2.1	_cmd_get	191
		5.29.2.2	cmd_get	193
		5.29.2.3	cmd_notify	194
		5.29.2.4	cmd_restart	194
		5.29.2.5	nprintf	195
		5.29.2.6	nprintind	196
		5.29.2.7	nprintkey	196
		5.29.2.8	restart	197
5.30	src/ckb	-daemon/r	notify.h File Reference	198
	5.30.1	Function	Documentation	199
		5.30.1.1	cmd_get	199
		5.30.1.2	cmd_notify	200
		5.30.1.3	cmd_restart	200
			nprintf	
		5.30.1.5	nprintind	201

XIV

5.30.1.6 nprintkey	12
5.31 src/ckb-daemon/os.h File Reference	
5.31.1 Macro Definition Documentation)3
5.31.1.1 _DEFAULT_SOURCE)3
5.31.1.2 _GNU_SOURCE)3
5.31.1.3 euid_guard_start)3
5.31.1.4 euid_guard_stop	3
5.31.1.5 UINPUT_VERSION)4
5.32 src/ckb-daemon/profile.c File Reference	14
5.32.1 Function Documentation	15
5.32.1.1 _freeprofile	15
5.32.1.2 allocprofile	15
5.32.1.3 cmd_erase	16
5.32.1.4 cmd_eraseprofile	17
5.32.1.5 cmd_id	17
5.32.1.6 cmd_name	8
5.32.1.7 cmd_profileid	8
5.32.1.8 cmd_profilename	19
5.32.1.9 freemode	19
5.32.1.10 freeprofile	0
5.32.1.11 gethwmodename	
5.32.1.12 gethwprofilename	1
5.32.1.13 getid	2
5.32.1.14 getmodename	2
5.32.1.15 getprofilename	3
5.32.1.16 hwtonative	4
5.32.1.17 initmode	4
5.32.1.18 loadprofile	5
5.32.1.19 nativetohw	
5.32.1.20 printname	6
5.32.1.21 setid	
5.32.1.22 u16dec	
5.32.1.23 u16enc	
5.32.1.24 urldecode2	
5.32.1.25 urlencode2	
5.32.2 Variable Documentation	
5.32.2.1 utf16to8	-
5.32.2.2 utf8to16	-
5.33 src/ckb-daemon/profile.h File Reference	
5.33.1 Macro Definition Documentation	1:1

CONTENTS xv

	5.33.1.1	hwloadprofile	. 221
5.33.	2 Function	Documentation	. 221
	5.33.2.1	allocprofile	. 221
	5.33.2.2	cmd_erase	. 222
	5.33.2.3	cmd_eraseprofile	. 223
	5.33.2.4	cmd_hwload_kb	. 223
	5.33.2.5	cmd_hwload_mouse	. 224
	5.33.2.6	cmd_hwsave_kb	. 225
	5.33.2.7	cmd_hwsave_mouse	. 226
	5.33.2.8	cmd_id	. 227
	5.33.2.9	cmd_name	. 227
	5.33.2.10	0 cmd_profileid	. 228
	5.33.2.11	1 cmd_profilename	. 228
	5.33.2.12	2 freeprofile	. 229
	5.33.2.13	3 gethwmodename	. 229
	5.33.2.14	4 gethwprofilename	. 230
	5.33.2.15	5 getid	. 231
	5.33.2.16	6 getmodename	. 231
	5.33.2.17	7 getprofilename	. 232
	5.33.2.18	8 hwtonative	. 233
	5.33.2.19	9 loadprofile	. 233
	5.33.2.20	0 nativetohw	. 234
	5.33.2.21	1 setid	. 234
5.34 src/cl	kb-daemon/	/profile_keyboard.c File Reference	. 235
5.34.	1 Function	Documentation	. 235
	5.34.1.1	cmd_hwload_kb	. 235
	5.34.1.2	cmd_hwsave_kb	. 236
	5.34.1.3	hwloadmode	. 237
5.35 src/cl	kb-daemon/	/profile_mouse.c File Reference	. 238
5.35.	1 Function	Documentation	. 238
	5.35.1.1	cmd_hwload_mouse	. 239
	5.35.1.2	cmd_hwsave_mouse	. 240
5.36 src/cl	kb-daemon/	structures.h File Reference	. 240
5.36.	1 Data Stru	ucture Documentation	. 242
	5.36.1.1	struct usbid	. 242
	5.36.1.2	struct macroaction	. 243
	5.36.1.3	struct keymacro	. 244
	5.36.1.4	struct binding	. 245
	5.36.1.5	struct dpiset	. 246
	5.36.1.6	struct lighting	. 247

xvi CONTENTS

	5.36.1.7 struct usbmode	
	5.36.1.8 struct usbprofile	
	5.36.1.9 struct hwprofile	
	5.36.1.10 struct usbinput	
	5.36.1.11 struct usbdevice	
5.36.2	Macro Definition Documentation	
	5.36.2.1 CLEAR_KEYBIT	
	5.36.2.2 DPI_COUNT	
	5.36.2.3 FEAT_ADJRATE	
	5.36.2.4 FEAT_ANSI	
	5.36.2.5 FEAT_BIND	
	5.36.2.6 FEAT_COMMON	
	5.36.2.7 FEAT_FWUPDATE	
	5.36.2.8 FEAT_FWVERSION	
	5.36.2.9 FEAT_HWLOAD	
	5.36.2.10 FEAT_ISO	
	5.36.2.11 FEAT_LMASK	
	5.36.2.12 FEAT_MONOCHROME	
	5.36.2.13 FEAT_MOUSEACCEL	
	5.36.2.14 FEAT_NOTIFY	
	5.36.2.15 FEAT_POLLRATE	
	5.36.2.16 FEAT_RGB	
	5.36.2.17 FEAT_STD_NRGB	
	5.36.2.18 FEAT_STD_RGB	
	5.36.2.19 HAS_ANY_FEATURE	
	5.36.2.20 HAS_FEATURES	
	5.36.2.21 HWMODE_K70	
	5.36.2.22 HWMODE_K95	
	5.36.2.23 HWMODE_MAX	
	5.36.2.24 I_CAPS	
	5.36.2.25 I_NUM	
	5.36.2.26 I_SCROLL	
	5.36.2.27 IFACE_MAX	
	5.36.2.28 KB_NAME_LEN	
	5.36.2.29 LIFT_MAX	
	5.36.2.30 LIFT_MIN	
	5.36.2.31 MACRO_MAX	
	5.36.2.32 MD_NAME_LEN	
	5.36.2.33 MODE_COUNT	
	5.36.2.34 MSG_SIZE	

CONTENTS xvii

		5.36.2.35	NEEDS_FW_UPDATE	 	 . 257
		5.36.2.36	OUTFIFO_MAX	 	 . 258
		5.36.2.37	PR_NAME_LEN	 	 . 258
		5.36.2.38	SCROLL_ACCELERATED	 	 . 258
		5.36.2.39	SCROLL_MAX	 	 . 258
		5.36.2.40	SCROLL_MIN	 	 . 258
		5.36.2.41	SERIAL_LEN	 	 . 258
		5.36.2.42	SET_KEYBIT	 	 . 258
	5.36.3	Variable [Documentation	 	 . 258
		5.36.3.1	vtable_keyboard	 	 . 258
		5.36.3.2	vtable_keyboard_nonrgb	 	 . 258
		5.36.3.3	vtable_mouse	 	 . 259
5.37	src/ckb	-daemon/u	usb.c File Reference	 	 . 259
	5.37.1	Function	Documentation	 	 . 260
		5.37.1.1	_resetusb	 	 . 260
		5.37.1.2	_setupusb	 	 . 260
		5.37.1.3	_usbrecv	 	 . 265
		5.37.1.4	_usbsend	 	 . 266
		5.37.1.5	closeusb	 	 . 267
		5.37.1.6	devmain	 	 . 269
		5.37.1.7	get_vtable	 	 . 271
		5.37.1.8	product_str	 	 . 272
		5.37.1.9	revertusb	 	 . 273
		5.37.1.10	setupusb	 	 . 274
		5.37.1.11	usb_tryreset	 	 . 275
		5.37.1.12	vendor_str	 	 . 276
	5.37.2	Variable [Documentation	 	 . 276
		5.37.2.1	features_mask	 	 . 276
		5.37.2.2	hwload_mode	 	 . 276
		5.37.2.3	reset_stop	 	 . 277
		5.37.2.4	usbmutex	 	 . 277
5.38	src/ckb	-daemon/ι	usb.h File Reference	 	 . 277
	5.38.1	Detailed I	Description	 	 . 281
	5.38.2	Macro De	efinition Documentation	 	 . 281
		5.38.2.1	DELAY_LONG	 	 . 281
		5.38.2.2	DELAY_MEDIUM	 	 . 281
		5.38.2.3	DELAY_SHORT	 	 . 281
		5.38.2.4	IS_FULLRANGE	 	 . 281
		5.38.2.5	IS_K65	 	 . 282
		5.38.2.6	IS_K70	 	 . 282

xviii CONTENTS

5.38.2.7 IS_K95
5.38.2.8 IS_M65
5.38.2.9 IS_MONOCHROME
5.38.2.10 IS_MONOCHROME_DEV
5.38.2.11 IS_MOUSE
5.38.2.12 IS_MOUSE_DEV
5.38.2.13 IS_RGB
5.38.2.14 IS_RGB_DEV
5.38.2.15 IS_SABRE
5.38.2.16 IS_SCIMITAR
5.38.2.17 IS_STRAFE
5.38.2.18 NK95_HWOFF
5.38.2.19 NK95_HWON
5.38.2.20 NK95_M1
5.38.2.21 NK95_M2
5.38.2.22 NK95_M3
5.38.2.23 nk95cmd
5.38.2.24 P_HARPOON
5.38.2.25 P_HARPOON_STR
5.38.2.26 P_K65
5.38.2.27 P_K65_LUX
5.38.2.28 P_K65_LUX_STR
5.38.2.29 P_K65_NRGB
5.38.2.30 P_K65_NRGB_STR
5.38.2.31 P_K65_RFIRE
5.38.2.32 P_K65_RFIRE_STR
5.38.2.33 P_K65_STR
5.38.2.34 P_K70
5.38.2.35 P_K70_LUX
5.38.2.36 P_K70_LUX_NRGB
5.38.2.37 P_K70_LUX_NRGB_STR
5.38.2.38 P_K70_LUX_STR
5.38.2.39 P_K70_NRGB
5.38.2.40 P_K70_NRGB_STR
5.38.2.41 P_K70_RFIRE
5.38.2.42 P_K70_RFIRE_NRGB
5.38.2.43 P_K70_RFIRE_NRGB_STR
5.38.2.44 P_K70_RFIRE_STR
5.38.2.45 P_K70_STR
5.38.2.46 P_K95

CONTENTS xix

	5.38.2.47 P_K95_NRGB	286
	5.38.2.48 P_K95_NRGB_STR	286
	5.38.2.49 P_K95_PLATINUM	286
	5.38.2.50 P_K95_PLATINUM_STR	286
	5.38.2.51 P_K95_STR	286
	5.38.2.52 P_M65	286
	5.38.2.53 P_M65_PRO	287
	5.38.2.54 P_M65_PRO_STR	287
	5.38.2.55 P_M65_STR	287
	5.38.2.56 P_SABRE_L	287
	5.38.2.57 P_SABRE_L_STR	287
	5.38.2.58 P_SABRE_N	287
	5.38.2.59 P_SABRE_N_STR	287
	5.38.2.60 P_SABRE_O	287
	5.38.2.61 P_SABRE_O2	287
	5.38.2.62 P_SABRE_O2_STR	287
	5.38.2.63 P_SABRE_O_STR	287
	5.38.2.64 P_SCIMITAR	288
	5.38.2.65 P_SCIMITAR_PRO	288
	5.38.2.66 P_SCIMITAR_PRO_STR	288
	5.38.2.67 P_SCIMITAR_STR	288
	5.38.2.68 P_STRAFE	288
	5.38.2.69 P_STRAFE_NRGB	288
	5.38.2.70 P_STRAFE_NRGB_STR	288
	5.38.2.71 P_STRAFE_STR	288
	5.38.2.72 resetusb	288
	5.38.2.73 USB_DELAY_DEFAULT	288
	5.38.2.74 usbrecv	288
	5.38.2.75 usbsend	289
	5.38.2.76 V_CORSAIR	289
	5.38.2.77 V_CORSAIR_STR	289
5.38.3	Function Documentation	289
	5.38.3.1 _nk95cmd	289
	5.38.3.2 _resetusb	291
	5.38.3.3 _usbrecv	291
	5.38.3.4 _usbsend	293
	5.38.3.5 closeusb	295
	5.38.3.6 os_closeusb	296
	5.38.3.7 os_inputmain	297
	5.38.3.8 os_resetusb	301

CONTENTS

	5.38.3.9 os_sendindicators
	5.38.3.10 os_setupusb
	5.38.3.11 os_usbrecv
	5.38.3.12 os_usbsend
	5.38.3.13 product_str
	5.38.3.14 revertusb
	5.38.3.15 setupusb
	5.38.3.16 usb_tryreset
	5.38.3.17 usbkill
	5.38.3.18 usbmain
	5.38.3.19 vendor_str
5.39 src/ckl	b-daemon/usb_linux.c File Reference
5.39.1	Data Structure Documentation
	5.39.1.1 struct _model
5.39.2	Macro Definition Documentation
	5.39.2.1 N_MODELS
	5.39.2.2 TEST_RESET
5.39.3	Function Documentation
	5.39.3.1 _nk95cmd
	5.39.3.2 os_closeusb
	5.39.3.3 os_inputmain
	5.39.3.4 os_resetusb
	5.39.3.5 os_sendindicators
	5.39.3.6 os_setupusb
	5.39.3.7 os_usbrecv
	5.39.3.8 os_usbsend
	5.39.3.9 strtrim
	5.39.3.10 udev_enum
	5.39.3.11 usb_add_device
	5.39.3.12 usb_rm_device
	5.39.3.13 usbadd
	5.39.3.14 usbclaim
	5.39.3.15 usbkill
	5.39.3.16 usbmain
	5.39.3.17 usbunclaim
5.39.4	Variable Documentation
	5.39.4.1 kbsyspath
	5.39.4.2 models
	5.39.4.3 udev
	5.39.4.4 udevthread

CONTENTS	 xxi
5.39.4.5 usbthread	 340
5.40 src/ckb-daemon/usb_mac.c File Reference	 340
Index	342

Chapter 1

Todo List

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

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

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

Global closeusb (usbdevice *kb)

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

Global devmain (usbdevice *kb)

Hope to find the need for dmutex usage later.

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

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

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

Global get_vtable (short vendor, short product)

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

Global os_inputmain (void *context)

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

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

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

Global os_setupusb (usbdevice *kb)

in these modules a pullrequest is outstanding

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

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

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

Global product_str (short product)

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

Global revertusb (usbdevice *kb)

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

2 Todo List

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

Global udevthread

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

Global udevthread

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

Global usb_add_device (struct udev_device *dev)

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

Global usb_tryreset (usbdevice *kb)

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

Global usbmain ()

Why isn't missing of uinput a fatal error?

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

Global usbmutex

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

Chapter 2

Data Structure Index

2.1	Data Structures	
Here	are the data structures with brief descriptions:	

devcmd.__unnamed__ 7

Data Structure Index

Chapter 3

File Index

3.1 File List

Here is a list of all files with br	ief descriptions
-------------------------------------	------------------

src/ckb-daemon/command.c
src/ckb-daemon/command.h
src/ckb-daemon/device.c
src/ckb-daemon/device.h
src/ckb-daemon/device_keyboard.c
src/ckb-daemon/device_mouse.c
src/ckb-daemon/device_vtable.c
src/ckb-daemon/devnode.c
src/ckb-daemon/devnode.h
src/ckb-daemon/dpi.c
src/ckb-daemon/dpi.h
src/ckb-daemon/extra_mac.c
src/ckb-daemon/firmware.c
src/ckb-daemon/firmware.h
src/ckb-daemon/includes.h
src/ckb-daemon/input.c
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
src/ckb-daemon/os.h
src/ckb-daemon/profile.c
src/ckb-daemon/profile.h
src/ckb-daemon/profile_keyboard.c
src/ckb-daemon/profile_mouse.c
src/ckb-daemon/structures.h
src/ckb-daemon/usb.c

6 File Index

src/ckb-daemon/usb.h	
Definitions for using USB interface	277
src/ckb-daemon/usb_linux.c	316
src/ckb-daemon/usb_mac.c	340

Chapter 4

Data Structure Documentation

4.1 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)

4.1.1 Detailed Description

Definition at line 78 of file command.h.

4.1.2 Field Documentation

4.1.2.1

4.1.2.2

4.1.2.3

4.1.2.4

4.1.2.5

4.1.2.6

4.1.2.7

4.1.2.8

4.1.2.9

4.1.2.10

4.1.2.11

4.1.2.12			
4.1.2.13			
4.1.2.14			
4.1.2.15			
4.1.2.16			
4.1.2.17			
4.1.2.18			
4.1.2.19			
4.1.2.20			
4.1.2.21			
4.1.2.22			
4.1.2.23			
4.1.2.24			
4.1.2.25			
4.1.2.26			
4.1.2.27			
4.1.2.28			
4.1.2.29			
4.1.2.30			
4.1.2.31			
4.1.2.32			
4.1.2.33			
4.1.2.34			
4.1.2.35			
4.1.2.36			

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



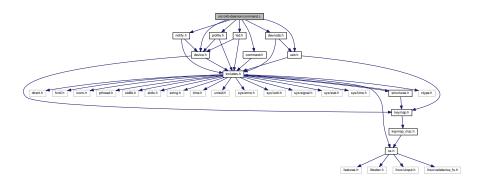
Chapter 5

File Documentation

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

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

Include dependency graph for command.c:



Macros

• #define TRY_WITH_RESET(action)

Functions

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

Variables

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

5.1.1 Macro Definition Documentation

12 File Documentation

5.1.1.1 #define TRY_WITH_RESET(action)

Value:

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

Definition at line 58 of file command.c.

Referenced by readcmd().

5.1.2 Function Documentation

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

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

Definition at line 67 of file command.c.

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

Referenced by devmain().

```
char* word = malloc(strlen(line) + 1);
68
69
       int wordlen;
       const char* newline = 0;
70
       const devcmd* vt = kb->vtable;
       usbprofile* profile = kb->profile;
73
       usbmode* mode = 0;
74
       int notifynumber = 0;
75
       // 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
80
           if(line > newline){
               mode = profile->currentmode;
command = NONE;
81
82
                notifynumber = 0;
                newline = strchr(line, ' \n');
                if(!newline)
85
86
                    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])){
                    command = i + CMD_FIRST;
92 #ifndef OS_MAC
                    // Layout and mouse acceleration aren't used on Linux; ignore
93
                    if (command == LAYOUT || command == ACCEL || command ==
94
      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
99
                              && command != ACTIVE && command != IDLE
100
                              && command != ERASE && command != ERASEPROFILE
```

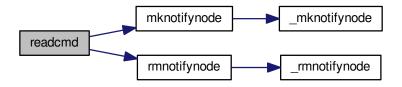
```
102
                              && command != RESTART)
103
                          goto next_loop;
104
                      break;
105
                 }
106
             }
107
108
             // Set current notification node when given @number
109
             int newnotify;
110
             if(sscanf(word, "@%u", &newnotify) == 1 && newnotify < OUTFIFO_MAX){</pre>
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
119
                 next_loop:
120
                 continue;
121
             // Reject anything not related to fwupdate if device has a bricked FW
122
              f(NEEDS_FW_UPDATE(kb) && command != FWUPDATE && command !=
123
      NOTIFYON && command != NOTIFYOFF)
                 continue;
121
125
126
             // Specially handled commands - these are available even when keyboard is IDLE
127
             switch (command) {
128
             case NOTIFYON: {
129
                 // Notification node on
130
                 int notify;
131
                 if(sscanf(word, "%u", &notify) == 1)
                 mknotifynode(kb, notify);
continue;
132
133
             } case NOTIFYOFF: {
134
135
                 // Notification node off
136
                 int notify;
137
                 if(sscanf(word, "%u", &notify) == 1 && notify != 0) // notify0 can't be removed
                     rmnotifynode(kb, notify);
138
                 continue;
139
140
             } case GET:
                 // Output data to notification node
141
142
                 vt->get(kb, mode, notifynumber, 0, word);
143
                 continue;
144
             case LAYOUT:
                 // OSX: switch ANSI/ISO keyboard layout
145
                 if(!strcmp(word, "ansi"))
146
                      kb->features = (kb->features & ~FEAT_LMASK) |
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
                // OSX mouse acceleration on/off
if(!strcmp(word, "on"))
   kb->features |= FEAT_MOUSEACCEL;
153
154
155
                 else if(!strcmp(word, "off"))
156
157
                     kb->features &= ~FEAT_MOUSEACCEL;
                 continue;
158
159
             case SCROLLSPEED:{
160
                int newscroll;
                 if(sscanf(word, "%d", &newscroll) != 1)
161
162
                     break:
                 if(newscroll < SCROLL_MIN)</pre>
163
                     newscroll = SCROLL_ACCELERATED;
164
                 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
178
             case FPS: {
                 // USB command delay (2 - 10ms)
179
180
                 uint framerate:
```

14 File Documentation

```
181
                 if(sscanf(word, "%u", &framerate) == 1 && framerate > 0) {
                      // Not all devices require the same number of messages per frame; select delay
182
       appropriately
                      uint per_frame = IS_MOUSE_DEV(kb) ? 2 : IS_FULLRANGE(kb) ? 14 : 5;
uint delay = 1000 / framerate / per_frame;
183
184
                      if (delay < 2)
185
                         delay = 2;
186
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){</pre>
196
                     kb->dither = dither;
197
                      profile->currentmode->light.forceupdate = 1;
198
199
                      mode->light.forceupdate = 1;
200
201
                 continue;
202
             case DELAY:
203
204
                 kb->delay = (!strcmp (word, "on")); // independendant from parameter to handle false
       commands like "delay off"
205
                 continue;
206
             case RESTART: {
                 char mybuffer[] = "no reason specified"; if (sscanf(line, " ^{n}_n", word) == -1) {
207
208
209
                      word = mvbuffer:
210
211
                 vt->do_cmd[command](kb, mode, notifynumber, 0, word);
212
                 continue;
213
             }
214
215
             default:;
216
217
218
             // If a keyboard is inactive, it must be activated before receiving any other commands
219
             if(!kb->active){
                 if (command == ACTIVE)
220
                     TRY_WITH_RESET(vt->active(kb, mode, notifynumber, 0, 0));
221
222
                 continue;
223
224
             // Specially handled commands only available when keyboard is ACTIVE
225
             switch(command) {
226
             case IDLE:
                 TRY_WITH_RESET(vt->idle(kb, mode, notifynumber, 0, 0));
227
228
                 continue;
             case SWITCH:
229
230
                 if(profile->currentmode != mode){
231
                      profile->currentmode = mode;
                      /// Set mode light for non-RGB K95
int index = INDEX_OF(mode, profile->mode);
232
233
                      vt->setmodeindex(kb, index);
234
235
                 }
236
                 continue;
237
             case HWLOAD: case HWSAVE:{
                 char delay = kb->usbdelay;
// Ensure delay of at least 10ms as the device can get overwhelmed otherwise
if(delay < 10)</pre>
238
239
240
241
                      kb->usbdelay = 10;
                  // Try to load/save the hardware profile. Reset on failure, disconnect if reset fails.
242
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)){
252
                      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
                 \ensuremath{//} Erase the current profile
                 vt->eraseprofile(kb, mode, notifynumber, 0, 0);
2.64
265
                 // Update profile/mode pointers
```

```
266
                profile = kb->profile;
267
                mode = profile->currentmode;
268
                 continue;
      case ERASE: case NAME: case IOFF: case ION: case IAUTO: case
INOTIFY: case PROFILENAME: case ID: case PROFILEID: case
269
      DPISEL: case LIFT: case SNAP:
270
                // All of the above just parse the whole word
271
                vt->do_cmd[command](kb, mode, notifynumber, 0, word);
272
                continue;
273
            case RGB: {
274
                // RGB command has a special response for a single hex constant
275
                int r, g, b;
if(sscanf(word, "%02x%02x%02x", &r, &g, &b) == 3){
276
                    // Set all keys
for(int i = 0; i < N_KEYS_EXTENDED; i++)</pre>
277
278
279
                         vt->rgb(kb, mode, notifynumber, i, word);
280
                     continue:
281
282
                break;
283
284
            case MACRO:
285
                if(!strcmp(word, "clear")){
                     // Macro has a special clear command
286
287
                     vt->macro(kb, mode, notifynumber, 0, 0);
288
                     continue;
289
290
                break;
291
            default:;
292
            // For anything else, split the parameter at the colon
293
            294
295
296
            if(left <= 0)
297
                continue;
            const char* right = word + left;
if(right[0] == ':')
298
299
300
                right++;
            // Macros and DPI have a separate left-side handler
301
302
            if(command == MACRO || command == DPI) {
303
                word[left] = 0;
304
                vt->do_macro[command](kb, mode, notifynumber, word, right);
305
                continue:
306
307
            // Scan the left side for key names and run the requested command
            int position = 0, field = 0;
308
309
            char keyname[11];
310
            while(position < left && sscanf(word + position, "%10[^:,]%n", keyname, &field) == 1){</pre>
311
                int keycode;
                 if(!strcmp(kevname, "all")){
312
313
                     // Set all keys
                     for (int i = 0; i < N_KEYS_EXTENDED; i++)</pre>
314
315
                         vt->do_cmd[command](kb, mode, notifynumber, i, right);
316
                } else if((sscanf(keyname, "#%d", &keycode) && keycode >= 0 && keycode <
      N_KEYS_EXTENDED)
317
                           || (sscanf(keyname, "#x%x", &keycode) && keycode >= 0 && keycode <
      N KEYS EXTENDED)) (
318
                     // Set a key numerically
319
                     vt->do_cmd[command](kb, mode, notifynumber, keycode, right);
320
                     321
322
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)){
   TRY_WITH_RESET(vt->updatergb(kb, 0));
335
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:



5.1.3 Variable Documentation

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

Definition at line 9 of file command.c.

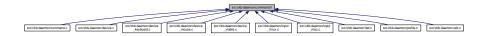
Referenced by readcmd().

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

#include "includes.h"
Include dependency graph for command.h:

colypa h down long house h systems 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

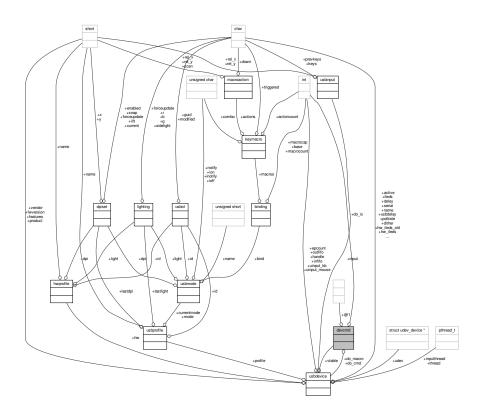
int readcmd (usbdevice *kb, const char *line)

5.2.1 Data Structure Documentation

5.2.1.1 union devcmd

Definition at line 73 of file command.h.

Collaboration diagram for devcmd:



Data Fields

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

5.2.2 Macro Definition Documentation

5.2.2.1 #define CMD_COUNT (CMD_LAST - CMD_FIRST + 2)

Definition at line 65 of file command.h.

Referenced by readcmd().

$5.2.2.2 \quad \hbox{\#define CMD_DEV_COUNT (CMD_LAST-CMD_VT_FIRST+1)}$

Definition at line 66 of file command.h.

5.2.3 Typedef Documentation

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

Definition at line 70 of file command.h.

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

Definition at line 71 of file command.h.

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

Definition at line 72 of file command.h.

- 5.2.3.4 typedef union devcmd devcmd
- 5.2.4 Enumeration Type Documentation
- 5.2.4.1 enum cmd

Enumerator

NONE

DELAY

CMD_FIRST

MODE

SWITCH

LAYOUT

ACCEL

SCROLLSPEED

NOTIFYON

NOTIFYOFF

FPS

DITHER

HWLOAD

CMD_VT_FIRST

HWSAVE

FWUPDATE

POLLRATE

ACTIVE

IDLE

ERASE

ERASEPROFILE

NAME

PROFILENAME

ID

PROFILEID

RGB

IOFF

ION

IAUTO

BIND

UNBIND

REBIND

MACRO

DPI

DPISEL

LIFT

SNAP

NOTIFY

INOTIFY

GET

RESTART

CMD_LAST

Definition at line 7 of file command.h.

```
// Special - handled by readcmd, no device functions NONE = -11,
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;
```

5.2.5 Function Documentation

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

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

Definition at line 67 of file command.c.

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

Referenced by devmain().

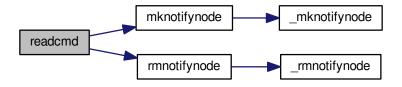
```
67
       char* word = malloc(strlen(line) + 1);
68
69
       int wordlen;
       const char* newline = 0;
70
       const devcmd* vt = kb->vtable;
       usbprofile* profile = kb->profile;
73
       usbmode* mode = 0;
       int notifynumber = 0;
74
75
       // Read words from the input
      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



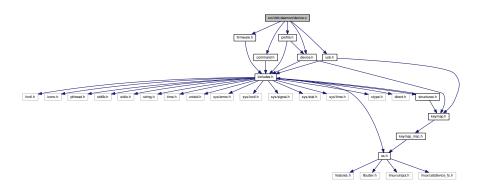
Here is the caller graph for this function:



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

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

Include dependency graph for device.c:



Functions

- int <u>_start_dev</u> (usbdevice *kb, int makeactive)
- int start_dev (usbdevice *kb, int makeactive)

Variables

- int hwload_mode = 1
- usbdevice keyboard [9]

hwload_mode = 1 means read hardware once. should be enough

pthread_mutex_t devlistmutex = PTHREAD_MUTEX_INITIALIZER

remember all usb devices. Needed for closeusb().

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

5.3.1 Function Documentation

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

start_dev get fw-info and pollrate; if available, install new firmware; get all hardware profiles_

Parameters

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

Returns

0 if success, other else

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

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

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

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

Why do not you just write it down?

- · Now check if device needs a firmware update. If so, set it up and leave the function without error.
- Device needs a firmware update. Finish setting up but don't do anything.
- Load profile from device if the hw-pointer is not set yet and hw-loading is possible and allowed.
 return error if mode == 2 (load always) and loading got an error. Else reset HWLOAD feature, because hwload must be 1.

That is real Horror code.

Definition at line 22 of file device.c.

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

Referenced by start dev().

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

```
44
                    kb->features &= ~(FEAT_FWVERSION |
      FEAT_FWUPDATE);
45
46
       if (NEEDS_FW_UPDATE (kb) ) {
51
           ckb_info("Device needs a firmware update. Please issue a fwupdate command.\n");
53
           kb->features = FEAT_RGB | FEAT_FWVERSION |
      FEAT_FWUPDATE;
55
          kb->active = 1;
56
           return 0;
57
       if(!kb->hw && hwload_mode && HAS_FEATURES(kb,
63
      FEAT_HWLOAD)) {
           if (hwloadprofile(kb, 1)) {
65
               if(hwload_mode == 2)
               return -1;
ckb_warn("Unable to load hardware profile\n");
66
67
               kb->features &= ~FEAT_HWLOAD;
68
          }
69
       // Active software mode if requested
72
       if (makeactive)
73
           return setactive(kb, 1);
74
       return 0;
75 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



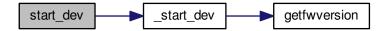
5.3.1.2 int start_dev (usbdevice * kb, int makeactive)

Definition at line 77 of file device.c.

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

```
77
78     // Force USB interval to 10ms during initial setup phase; return to nominal 5ms after setup completes.
79     kb->usbdelay = 10;
80     int res = _start_dev(kb, makeactive);
81     kb->usbdelay = USB_DELAY_DEFAULT;
82     return res;
83 }
```

Here is the call graph for this function:



5.3.2 Variable Documentation

5.3.2.1 pthread_mutex_t devlistmutex = PTHREAD_MUTEX_INITIALIZER

Definition at line 11 of file device.c.

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

Definition at line 12 of file device.c.

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

5.3.2.3 int hwload_mode = 1

Definition at line 7 of file device.c.

Referenced by start dev(), usbrecv(), usbsend(), and main().

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

Definition at line 13 of file device.c.

5.3.2.5 usbdevice keyboard[9]

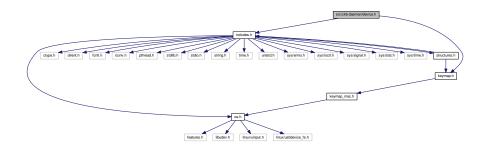
Definition at line 10 of file device.c.

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

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

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

Include dependency graph for device.h:



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



Macros

- #define DEV_MAX 9
- #define IS_CONNECTED(kb) ((kb) && (kb)->handle && (kb)->uinput_kb && (kb)->uinput_mouse)
- #define dmutex(kb) (devmutex + INDEX_OF(kb, keyboard))
- #define imutex(kb) (inputmutex + INDEX_OF(kb, keyboard))
- #define setactive(kb, makeactive) ((makeactive) ? (kb)->vtable->active((kb), 0, 0, 0, 0) : (kb)->vtable->idle((kb), 0, 0, 0, 0))

setactive() calls via the corresponding kb->vtable either the active() or the idle() function. active() is called if the parameter makeactive is true, idle if it is false.

What function is called effectively is device dependent. Have a look at device_vtable.c for more information.

- #define IN HID 0x80
- #define IN CORSAIR 0x40
- #define ACT_LIGHT 1
- #define ACT_NEXT 3
- #define ACT_NEXT_NOWRAP 5
- #define ACT_LOCK 8
- #define ACT MR RING 9
- #define ACT M1 10
- #define ACT_M2 11
- #define ACT_M3 12

Functions

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

Variables

• usbdevice keyboard [9]

hwload_mode = 1 means read hardware once. should be enough

- pthread_mutex_t devmutex [9]
- pthread_mutex_t inputmutex [9]

5.4.1 Macro Definition Documentation

5.4.1.1 #define ACT_LIGHT 1

Definition at line 60 of file device.h.

Referenced by setactive_kb().

5.4.1.2 #define ACT_LOCK 8

Definition at line 63 of file device.h.

Referenced by setactive_kb().

5.4.1.3 #define ACT_M1 10

Definition at line 65 of file device.h.

Referenced by setactive_kb().

5.4.1.4 #define ACT_M2 11

Definition at line 66 of file device.h.

Referenced by setactive_kb().

5.4.1.5 #define ACT_M3 12

Definition at line 67 of file device.h.

Referenced by setactive_kb().

5.4.1.6 #define ACT_MR_RING 9

Definition at line 64 of file device.h.

Referenced by setactive_kb().

5.4.1.7 #define ACT_NEXT 3

Definition at line 61 of file device.h.

5.4.1.8 #define ACT_NEXT_NOWRAP 5

Definition at line 62 of file device.h.

5.4.1.9 #define DEV_MAX 9

Definition at line 8 of file device.h.

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

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

Definition at line 18 of file device.h.

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

5.4.1.11 #define imutex(kb) (inputmutex + INDEX OF(kb, keyboard))

Definition at line 22 of file device.h.

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

5.4.1.12 #define IN_CORSAIR 0x40

Definition at line 57 of file device.h.

Referenced by setactive kb(), and setactive mouse().

5.4.1.13 #define IN_HID 0x80

Definition at line 56 of file device.h.

Referenced by setactive_kb(), and setactive_mouse().

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

Definition at line 12 of file device.h.

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

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

Definition at line 36 of file device.h.

Referenced by start dev(), and revertusb().

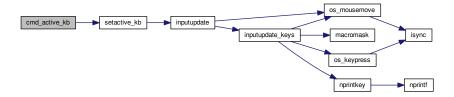
5.4.2 Function Documentation

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

Definition at line 112 of file device keyboard.c.

References setactive_kb().

Here is the call graph for this function:



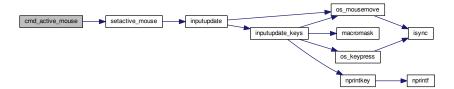
5.4.2.2 int cmd_active_mouse (usbdevice * kb, usbmode * dummy1, int dummy2, int dummy3, const char * dummy4)

Definition at line 44 of file device_mouse.c.

References setactive_mouse().

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

Here is the call graph for this function:



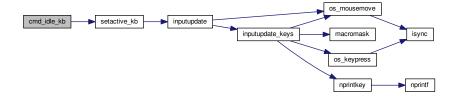
5.4.2.3 int cmd_idle_kb (usbdevice * kb, usbmode * dummy1, int dummy2, int dummy3, const char * dummy4)

Definition at line 116 of file device_keyboard.c.

References setactive_kb().

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

Here is the call graph for this function:



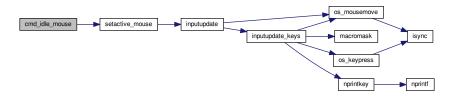
5.4.2.4 int cmd_idle_mouse (usbdevice * kb, usbmode * dummy1, int dummy2, int dummy3, const char * dummy4)

Definition at line 48 of file device_mouse.c.

References setactive_mouse().

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

Here is the call graph for this function:



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

Definition at line 52 of file device mouse.c.

References MSG SIZE, usbdevice::pollrate, and usbsend.

5.4.2.6 int setactive_kb (usbdevice * kb, int active)

Definition at line 18 of file device_keyboard.c.

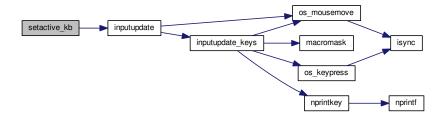
References ACT_LIGHT, ACT_LOCK, ACT_M1, ACT_M2, ACT_M3, ACT_MR_RING, usbdevice::active, DELAY_MEDIUM, lighting::forceupdate, imutex, IN_CORSAIR, IN_HID, usbdevice::input, inputupdate(), keymap, usbinput::keys, usbprofile::lastlight, MSG_SIZE, N_KEYS_HW, NEEDS_FW_UPDATE, usbdevice::profile, usbsend, and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

Referenced by cmd_active_kb(), and cmd_idle_kb().

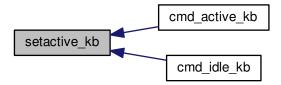
```
18
19
       if (NEEDS_FW_UPDATE(kb))
           return 0;
22
       pthread_mutex_lock(imutex(kb));
23
       kb->active = !!active;
       kb->profile->lastlight.forceupdate = 1;
24
       // Clear input
26
       memset(&kb->input.keys, 0, sizeof(kb->input.keys));
       inputupdate(kb);
28
 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

```
33
            { 0x07, 0x05, 2, 0, 0x03, 0x00 } // Commits key input selection
34
        if (active) {
35
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;
40
            DELAY_MEDIUM(kb);
            // Set input mode on the keys. They must be grouped into packets of 60 bytes (+ 4 bytes header)
// Keys are referenced in byte pairs, with the first byte representing the key and the second byte
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
                     // Enable hardware actions
78
                     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
                 msq[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
         // 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:



5.4.2.7 int setactive_mouse (usbdevice * kb, int active)

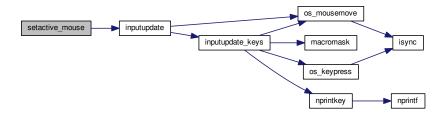
Definition at line 9 of file device_mouse.c.

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

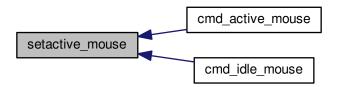
Referenced by cmd_active_mouse(), and cmd_idle_mouse().

```
10
       if (NEEDS_FW_UPDATE(kb))
            return 0;
12
       const int keycount = 20;
       uchar msg[2][MSG_SIZE] = {
    { 0x07, 0x04, 0 },
    { 0x07, 0x40, keycount, 0 },
13
                                                     // 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
           // Restore HW mode
21
            msg[0][2] = 1;
23
       pthread_mutex_lock(imutex(kb));
24
       kb->active = !!active;
       kb->profile->lastlight.forceupdate = 1;
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
            if(!usbsend(kb, msg[1], 1))
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.4.2.8 void setmodeindex_nrgb (usbdevice * kb, int index)

Definition at line 120 of file device_keyboard.c.

References NK95_M1, NK95_M2, NK95_M3, and nk95cmd.

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

5.4.2.9 int start_dev (usbdevice * kb, int makeactive)

Definition at line 77 of file device.c.

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

Here is the call graph for this function:



5.4.2.10 int start_kb_nrgb (usbdevice * kb, int makeactive)

Definition at line 9 of file device_keyboard.c.

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

5.4.3 Variable Documentation

5.4.3.1 pthread_mutex_t devmutex[9]

Definition at line 12 of file device.c.

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

5.4.3.2 pthread_mutex_t inputmutex[9]

Definition at line 13 of file device.c.

5.4.3.3 usbdevice keyboard[9]

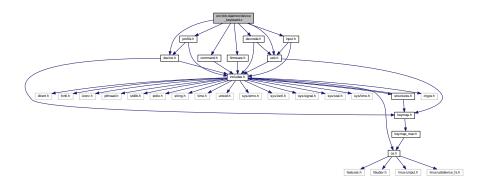
Definition at line 10 of file device.c.

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

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

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

Include dependency graph for device_keyboard.c:



Functions

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

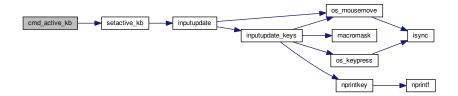
5.5.1 Function Documentation

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

Definition at line 112 of file device_keyboard.c.

References setactive_kb().

Here is the call graph for this function:



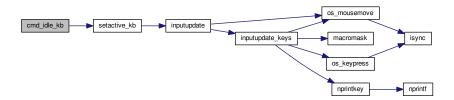
5.5.1.2 int cmd_idle_kb (usbdevice * kb, usbmode * dummy1, int dummy2, int dummy3, const char * dummy4)

Definition at line 116 of file device_keyboard.c.

References setactive_kb().

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

Here is the call graph for this function:



5.5.1.3 int setactive_kb (usbdevice * kb, int active)

Definition at line 18 of file device_keyboard.c.

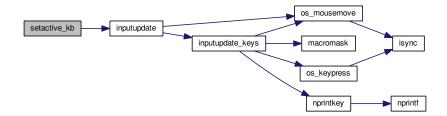
References ACT_LIGHT, ACT_LOCK, ACT_M1, ACT_M2, ACT_M3, ACT_MR_RING, usbdevice::active, DELAY_MEDIUM, lighting::forceupdate, imutex, IN_CORSAIR, IN_HID, usbdevice::input, inputupdate(), keymap, usbinput::keys, usbprofile::lastlight, MSG_SIZE, N_KEYS_HW, NEEDS_FW_UPDATE, usbdevice::profile, usbsend, and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

Referenced by cmd_active_kb(), and cmd_idle_kb().

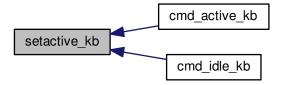
```
18
       if (NEEDS_FW_UPDATE(kb))
19
20
            return 0;
21
       pthread_mutex_lock(imutex(kb));
22
        kb->active = !!active;
       kb->profile->lastlight.forceupdate = 1;
25
       // Clear input
26
       memset(&kb->input.keys, 0, sizeof(kb->input.keys));
27
       inputupdate(kb);
28
       pthread_mutex_unlock(imutex(kb));
       uchar msg[3][MSG_SIZE] = {
30
31
            \{ 0x07, 0x04, 0 \},
                                                    // Disables or enables {\tt HW} control for top row
32
            \{ 0x07, 0x40, 0 \},
                                                    // Selects key input
            \{ 0x07, 0x05, 2, 0, 0x03, 0x00 \}
                                                    // Commits key input selection
33
34
35
       if (active) {
36
            // Put the M-keys (K95) as well as the Brightness/Lock keys into software-controlled mode.
37
            msg[0][2] = 2;
38
            if(!usbsend(kb, msg[0], 1))
39
                return -1:
           DELAY_MEDIUM(kb);
40
            // Set input mode on the keys. They must be grouped into packets of 60 bytes (+ 4 bytes header)
41
42
            // Keys are referenced in byte pairs, with the first byte representing the key and the second byte
43
            for(int key = 0; key < N_KEYS_HW; ){</pre>
                int pair;
for(pair = 0; pair < 30 && key < N_KEYS_HW; pair++, key++){</pre>
44
45
46
                     // Select both standard and Corsair input. The standard input will be ignored except in
                     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;
50
51
                     msg[1][5 + pair * 2] = action;
```

```
53
                // Byte 2 = pair count (usually 30, less on final message)
msg[1][2] = pair;
55
                if(!usbsend(kb, msg[1], 1))
56
57
                    return -1;
58
            // Commit new input settings
59
60
            if(!usbsend(kb, msg[2], 1))
61
                return -1;
62
           DELAY_MEDIUM(kb);
63
       } else {
           // Set the M-keys back into hardware mode, restore hardware RGB profile. It has to be sent twice
64
       for some reason.
65
           msg[0][2] = 1;
66
            if(!usbsend(kb, msg[0], 1))
           return -1;
DELAY_MEDIUM(kb);
67
68
            if(!usbsend(kb, msg[0], 1))
69
70
                return -1;
            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.
for(int key = 0; key < N_KEYS_HW; ){</pre>
74
75
                int pair;
                for(pair = 0; pair < 30 && key < N_KEYS_HW; pair++, key++){</pre>
76
77
                    uchar action = IN_HID;
78
                     // Enable hardware actions
79
                     if(keymap[key].name){
                         if(!strcmp(keymap[key].name, "mr"))
    action = ACT_MR_RING;
80
81
82
                         else if(!strcmp(keymap[key].name, "m1"))
83
                             action = ACT_M1;
84
                         else if(!strcmp(keymap[key].name, "m2"))
85
                             action = ACT_M2;
                         else if(!strcmp(keymap[key].name, "m3"))
86
                             action = ACT M3;
87
88
                         else if(!strcmp(keymap[key].name, "light"))
89
                             action = ACT_LIGHT;
90
                         else if(!strcmp(keymap[key].name, "lock"))
91
                             action = ACT_LOCK;
92
                    msg[1][4 + pair * 2] = key;
93
                    msg[1][5 + pair * 2] = action;
94
96
                // Byte 2 = pair count (usually 30, less on final message)
                msg[1][2] = pair;
97
98
                if(!usbsend(kb, msg[1], 1))
99
                    return -1:
100
101
             // Commit new input settings
102
             if(!usbsend(kb, msg[2], 1))
103
                 return -1;
104
             DELAY_MEDIUM(kb);
105 #endif
106
107
         .
// Update indicator LEDs if the profile contains settings for them
108
        kb->vtable->updateindicators(kb, 0);
109
        return 0;
110 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.5.1.4 void setmodeindex_nrgb (usbdevice * kb, int index)

Definition at line 120 of file device_keyboard.c.

References NK95_M1, NK95_M2, NK95_M3, and nk95cmd.

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

5.5.1.5 int start_kb_nrgb (usbdevice * kb, int makeactive)

Definition at line 9 of file device_keyboard.c.

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

```
// Put the non-RGB K95 into software mode. Nothing else needs to be done hardware wise nk95cmd(kb, NK95_HWOFF);

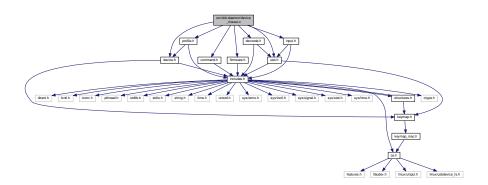
// Fill out RGB features for consistency, even though the keyboard doesn't have them kb->active = 1;

kb->pollrate = -1;
return 0;
```

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

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

Include dependency graph for device_mouse.c:



Functions

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

5.6.1 Function Documentation

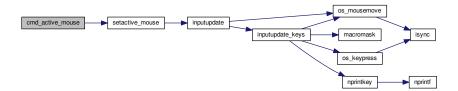
5.6.1.1 int cmd_active_mouse (usbdevice * kb, usbmode * dummy1, int dummy2, int dummy3, const char * dummy4)

Definition at line 44 of file device_mouse.c.

References setactive_mouse().

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

Here is the call graph for this function:



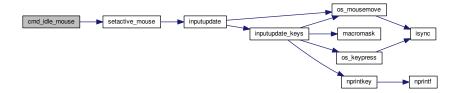
5.6.1.2 int cmd_idle_mouse (usbdevice * kb, usbmode * dummy1, int dummy2, int dummy3, const char * dummy4)

Definition at line 48 of file device_mouse.c.

References setactive_mouse().

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

Here is the call graph for this function:



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

Definition at line 52 of file device mouse.c.

References MSG_SIZE, usbdevice::pollrate, and usbsend.

```
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 }
```

5.6.1.4 int setactive_mouse (usbdevice * kb, int active)

Definition at line 9 of file device mouse.c.

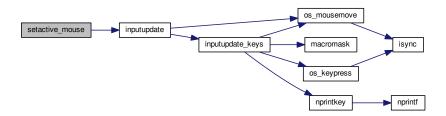
References usbdevice::active, lighting::forceupdate, imutex, IN_CORSAIR, IN_HID, usbdevice::input, inputupdate(), usbinput::keys, usbprofile::lastlight, MSG_SIZE, NEEDS_FW_UPDATE, usbdevice::profile, and usbsend.

Referenced by cmd active mouse(), and cmd idle mouse().

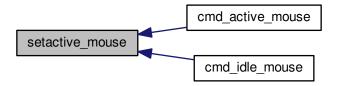
```
if (NEEDS_FW_UPDATE (kb) )
10
11
           return 0;
       const int keycount = 20;
12
13
       uchar msg[2][MSG_SIZE] = {
          \{ 0x07, 0x04, 0 \},
                                                  // Disables or enables {\tt HW} control for DPI and Sniper button
           { 0x07, 0x40, keycount, 0 },
                                                  // Select button input (simlilar to the packet sent to
15
       keyboards, but lacks a commit packet)
16
       if(active)
17
18
           // Put the mouse into SW mode
19
           msg[0][2] = 2;
20
           // Restore HW mode
2.1
22
           msg[0][2] = 1;
       pthread_mutex_lock(imutex(kb));
23
       kb->active = !!active;
24
       kb->profile->lastlight.forceupdate = 1;
25
       // Clear input
       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;
       if(active){
32
33
           // Set up key input
34
           if(!usbsend(kb, msg[1], 1))
           return -1;
for(int i = 0; i < keycount; i++) {</pre>
3.5
36
37
               msg[1][i * 2 + 4] = i + 1;
               msg[1][i * 2 + 5] = (i < 6 ? IN_HID : IN_CORSAIR);
```

```
39 }
40 }
41 return 0;
42 }
```

Here is the call graph for this function:



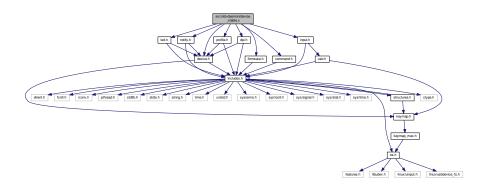
Here is the caller graph for this function:



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

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

Include dependency graph for device_vtable.c:



Functions

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

Variables

- · const devcmd vtable keyboard
- const devcmd vtable_keyboard_nonrgb
- · const devcmd vtable_mouse

5.7.1 Function Documentation

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

Definition at line 13 of file device vtable.c.

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

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

Definition at line 16 of file device_vtable.c.

```
16
{
```

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

Definition at line 11 of file device_vtable.c.

```
11
12 }
```

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

Definition at line 23 of file device_vtable.c.

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

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

Definition at line 21 of file device_vtable.c.

```
21 {
```

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

Definition at line 18 of file device_vtable.c.

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

5.7.2 Variable Documentation

5.7.2.1 const devcmd vtable_keyboard

Definition at line 28 of file device vtable.c.

Referenced by get_vtable().

5.7.2.2 const devcmd vtable_keyboard_nonrgb

Definition at line 75 of file device_vtable.c.

Referenced by get_vtable().

5.7.2.3 const devcmd vtable_mouse

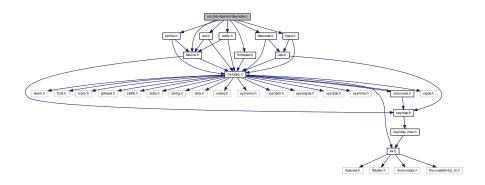
Definition at line 122 of file device_vtable.c.

Referenced by get_vtable().

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

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

Include dependency graph for devnode.c:



Data Structures

· struct _readlines_ctx

Macros

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

Functions

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

_updateconnected Update the list of connected devices.

void updateconnected ()

Update the list of connected devices.

- int _mknotifynode (usbdevice *kb, int notify)
- int mknotifynode (usbdevice *kb, int notify)

Creates a notification node for the specified keyboard.

- int _rmnotifynode (usbdevice *kb, int notify)
- int rmnotifynode (usbdevice *kb, int notify)

Removes a notification node for the specified keyboard.

- static int mkdevpath (usbdevice *kb)
- int mkdevpath (usbdevice *kb)

Create a dev path for the keyboard at index. Returns 0 on success.

int rmdevpath (usbdevice *kb)

Remove the dev path for the keyboard at index. Returns 0 on success.

• int mkfwnode (usbdevice *kb)

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

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

Variables

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

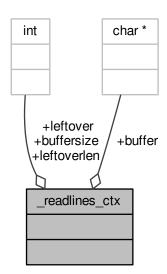
Group ID for the control nodes. -1 to give read/write access to everybody.

5.8.1 Data Structure Documentation

5.8.1.1 struct _readlines_ctx

Definition at line 335 of file devnode.c.

Collaboration diagram for _readlines_ctx:



Data Fields

char *	buffer	
int	buffersize	
int	leftover	
int	leftoverlen	

5.8.2 Macro Definition Documentation

5.8.2.1 #define MAX_BUFFER (1024 * 1024 - 1)

Definition at line 334 of file devnode.c.

Referenced by readlines().

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

Definition at line 17 of file devnode.c.

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

5.8.3 Function Documentation

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

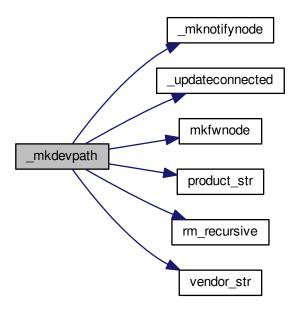
Definition at line 136 of file devnode.c.

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

Referenced by mkdevpath().

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

```
190
                        | | (kb->infifo = open(inpath, O_RDWR) + 1) == 0) {
                   // Add one to the FD because 0 is a valid descriptor, but ckb uses 0 for uninitialized devices ckb_err("Unable to create %s: %s\n", inpath, strerror(errno));
191
192
193
                   rm_recursive(path);
194
                   kb->infifo = 0;
195
                   return -1:
196
197
              if(gid >= 0)
198
                   fchown(kb->infifo - 1, 0, gid);
199
200
              // Create notification FIFO
201
              _mknotifynode(kb, 0);
202
203
              // Write the model and serial to files
204
              char mpath[sizeof(path) + 6], spath[sizeof(path) + 7];
              snprintf(mpath, sizeof(mpath), "%s/model", path);
snprintf(spath, sizeof(spath), "%s/serial", path);
FILE* mfile = fopen(mpath, "w");
205
206
207
              if(mfile) {
208
209
                   fputs(kb->name, mfile);
210
                   fputc(' \ n', mfile);
211
                   fclose(mfile);
                   chmod(mpath, S_GID_READ);
212
                   if(gid >= 0)
213
214
                       chown (mpath, 0, gid);
              } else {
215
216
                   ckb_warn("Unable to create %s: %sn", mpath, strerror(errno));
217
                   remove(mpath);
218
219
              FILE* sfile = fopen(spath, "w");
220
              if(sfile){
221
                   fputs(kb->serial, sfile);
222
                   fputc('\n', sfile);
223
                   fclose(sfile);
224
                   chmod(spath, S_GID_READ);
225
                   if(gid >= 0)
226
                       chown(spath, 0, gid);
              } else {
227
228
                   ckb_warn("Unable to create %s: %s\n", spath, strerror(errno));
229
                   remove(spath);
230
              // Write the keyboard's features
2.31
              char fpath[sizeof(path) + 9];
232
              snprintf(fpath, sizeof(fpath), "%s/features", path);
233
234
              FILE* ffile = fopen(fpath, "w");
235
              if(ffile){
236
                   fprintf(ffile, "%s %s", vendor_str(kb->vendor),
       product_str(kb->product));
    if(HAS_FEATURES(kb, FEAT_MONOCHROME))
237
                   fputs(" monochrome", ffile);
if(HAS_FEATURES(kb, FEAT_RGB))
238
239
240
                        fputs(" rgb", ffile);
241
                   if(HAS_FEATURES(kb, FEAT_POLLRATE))
                   fputs(" pollrate", ffile);
if(HAS_FEATURES(kb, FEAT_ADJRATE))
242
243
                   fputs(" adjrate", ffile);
if(HAS_FEATURES(kb, FEAT_BIND))
244
245
246
                        fputs(" bind", ffile);
                   if(HAS_FEATURES(kb, FEAT_NOTIFY))
  fputs(" notify", ffile);
if(HAS_FEATURES(kb, FEAT_FWVERSION))
247
248
249
                        fputs(" fwversion", ffile);
250
251
                   if(HAS_FEATURES(kb, FEAT_FWUPDATE))
                   fputs(" fwupdate", ffile);
fputc('\n', ffile);
252
253
                   fclose(ffile);
254
255
                   chmod(fpath, S_GID_READ);
256
                   if(gid >= 0)
257
                       chown(fpath, 0, gid);
258
              } else {
259
                   ckb_warn("Unable to create %s: %sn", fpath, strerror(errno));
260
                   remove(fpath);
261
              // Write firmware version and poll rate
262
              mkfwnode(kb);
263
264
265
          return 0;
266 }
```



Here is the caller graph for this function:



5.8.3.2 int _mknotifynode (usbdevice * kb, int notify)

Definition at line 87 of file devnode.c.

References ckb_warn, devpath, gid, INDEX_OF, keyboard, usbdevice::outfifo, OUTFIFO_MAX, and S_GID_READ. Referenced by _mkdevpath(), and mknotifynode().

```
87
          if (notify < 0 || notify >= OUTFIFO_MAX)
88
89
                 return -1;
90
          if(kb->outfifo[notify] != 0)
91
                return 0;
          // Create the notification node
int index = INDEX_OF(kb, keyboard);
92
93
        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){
94
96
                // 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));
97
98
                kb->outfifo[notify] = 0;
99
100
                 remove(outpath);
101
                  return -1;
102
            if(gid >= 0)
103
104
                  fchown(kb->outfifo[notify] - 1, 0, gid);
105
            return 0;
106 }
```

Here is the caller graph for this function:



5.8.3.3 int rmnotifynode (usbdevice * kb, int notify)

Definition at line 115 of file devnode.c.

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

Referenced by rmdevpath(), and rmnotifynode().

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

Here is the caller graph for this function:



5.8.3.4 void updateconnected ()

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

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

eg

/dev/input/ckb1 0F022014ABABABABABABABABABABABA999 Corsair K95 RGB Gaming Keyboard

/dev/input/ckb2 0D02303DBACBACBACBACBACBACBACBACBAC998 Corsair M65 RGB Gaming Mouse

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

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

Definition at line 55 of file devnode.c.

References ckb_warn, DEV_MAX, devmutex, devpath, gid, IS_CONNECTED, keyboard, and S_GID_READ. Referenced by _mkdevpath(), and updateconnected().

55 {

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



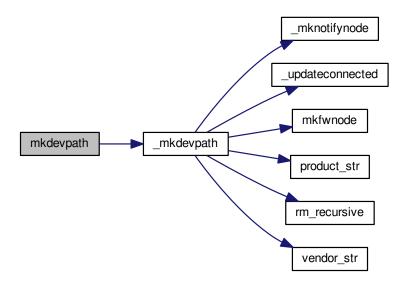
5.8.3.5 int mkdevpath (usbdevice * kb)

Definition at line 268 of file devnode.c.

References _mkdevpath(), euid_guard_start, and euid_guard_stop.

Referenced by _setupusb(), and main().

Here is the call graph for this function:



Here is the caller graph for this function:



5.8.3.6 int mkfwnode (usbdevice * kb)

Definition at line 299 of file devnode.c.

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

Referenced by _mkdevpath(), and fwupdate().

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

```
319
        if (pfile) {
320
            fprintf(pfile, "%d ms", kb->pollrate);
            fputc('\n', pfile);
fclose(pfile);
321
322
           chmod(ppath, S_GID_READ);
if(gid >= 0)
323
324
325
                chown(ppath, 0, gid);
326
327
           ckb_warn("Unable to create %s: %s\n", fwpath, strerror(errno));
328
            remove(ppath);
329
            return -2;
330
331
        return 0;
332 }
```



5.8.3.7 int mknotifynode (usbdevice * kb, int notify)

Definition at line 108 of file devnode.c.

References _mknotifynode(), euid_guard_start, and euid_guard_stop.

Referenced by readcmd().

Here is the call graph for this function:



Here is the caller graph for this function:



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

Definition at line 353 of file devnode.c.

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

Referenced by devmain().

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

Here is the caller graph for this function:



5.8.3.9 void readlines_ctx_free (readlines_ctx ctx)

Definition at line 348 of file devnode.c.

References _readlines_ctx::buffer.

Referenced by devmain().

348 {

```
readlines_ctx_free devinain selepusb usbadd device usb_add_device usb_add_device usb_add_device usb_add_device
```

5.8.3.10 void readlines_ctx_init (readlines_ctx * ctx)

Definition at line 341 of file devnode.c.

Referenced by devmain().

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

Here is the caller graph for this function:



5.8.3.11 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)
22
            return remove(path);
       struct dirent* file;
while((file = readdir(dir)))
23
2.4
25
26
            if(!strcmp(file->d_name, ".") || !strcmp(file->d_name, ".."))
            char path2[FILENAME_MAX];
28
            snprintf(path2, FILENAME_MAX, "%s/%s", path, file->d_name);
29
30
            int stat = rm_recursive(path2);
if(stat != 0)
31
                return stat;
32
33
34
       closedir(dir);
35
       return remove(path);
36 }
```

Here is the caller graph for this function:



5.8.3.12 int rmdevpath (usbdevice * kb)

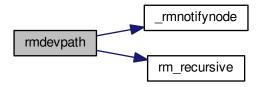
Definition at line 275 of file devnode.c.

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

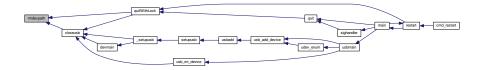
Referenced by closeusb(), and quitWithLock().

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

Here is the call graph for this function:



Here is the caller graph for this function:



5.8.3.13 int rmnotifynode (usbdevice * kb, int notify)

Definition at line 129 of file devnode.c.

References _rmnotifynode(), euid_guard_start, and euid_guard_stop.

Referenced by readcmd().

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



Here is the caller graph for this function:

```
muni yobu usba da dayana d
```

5.8.3.14 void updateconnected ()

Definition at line 81 of file devnode.c.

References _updateconnected(), euid_guard_start, and euid_guard_stop.

Referenced by _setupusb(), and closeusb().

Here is the call graph for this function:



Here is the caller graph for this function:



5.8.4 Variable Documentation

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

Definition at line 11 of file devnode.c.

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

5.8.4.2 long gid = -1

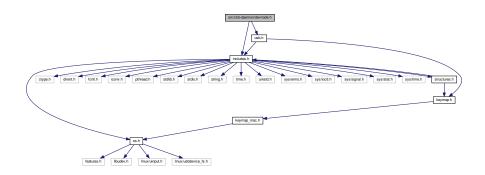
Definition at line 16 of file devnode.c.

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

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

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

Include dependency graph for devnode.h:



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



Macros

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

Typedefs

• typedef struct _readlines_ctx * readlines_ctx

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

Functions

· void updateconnected ()

Update the list of connected devices.

• int mkdevpath (usbdevice *kb)

Create a dev path for the keyboard at index. Returns 0 on success.

int rmdevpath (usbdevice *kb)

Remove the dev path for the keyboard at index. Returns 0 on success.

• int mknotifynode (usbdevice *kb, int notify)

Creates a notification node for the specified keyboard.

int rmnotifynode (usbdevice *kb, int notify)

Removes a notification node for the specified keyboard.

int mkfwnode (usbdevice *kb)

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

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

Variables

· const char *const devpath

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

long gid

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

5.9.1 Macro Definition Documentation

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

Definition at line 17 of file devnode.h.

Referenced by _mkdevpath().

5.9.1.2 #define S_CUSTOM_R (S_IRUSR | S_IWUSR | S_IRGRP)

Definition at line 18 of file devnode.h.

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

Definition at line 15 of file devnode.h.

Referenced by _mkdevpath().

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

Definition at line 14 of file devnode.h.

Referenced by _mkdevpath().

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

Definition at line 16 of file devnode.h.

Referenced by _mkdevpath().

5.9.2 Typedef Documentation

5.9.2.1 typedef struct _readlines_ctx* readlines_ctx

Definition at line 39 of file devnode.h.

5.9.3 Function Documentation

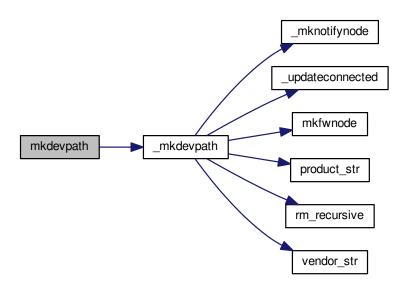
5.9.3.1 int mkdevpath (usbdevice * kb)

Definition at line 268 of file devnode.c.

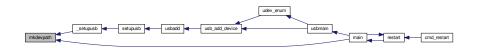
References _mkdevpath(), euid_guard_start, and euid_guard_stop.

Referenced by _setupusb(), and main().

Here is the call graph for this function:



Here is the caller graph for this function:



5.9.3.2 int mkfwnode (usbdevice * kb)

Definition at line 299 of file devnode.c.

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

Referenced by _mkdevpath(), and fwupdate().

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

Here is the caller graph for this function:



5.9.3.3 int mknotifynode (usbdevice * kb, int notify)

Definition at line 108 of file devnode.c.

References _mknotifynode(), euid_guard_start, and euid_guard_stop.

Referenced by readcmd().

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

Here is the call graph for this function:



Here is the caller graph for this function:



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

Definition at line 353 of file devnode.c.

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

Referenced by devmain().

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



5.9.3.5 void readlines_ctx_free (readlines_ctx ctx)

Definition at line 348 of file devnode.c.

References _readlines_ctx::buffer.

Referenced by devmain().

Here is the caller graph for this function:

```
readines, ctx, free devmain setupusb setupusb description usbadd device usb add d
```

5.9.3.6 void readlines_ctx_init (readlines_ctx * ctx)

Definition at line 341 of file devnode.c.

Referenced by devmain().

Here is the caller graph for this function:



5.9.3.7 int rmdevpath (usbdevice * kb)

Definition at line 275 of file devnode.c.

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

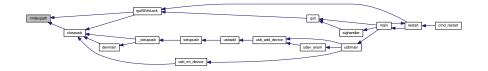
Referenced by closeusb(), and quitWithLock().

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

Here is the call graph for this function:



Here is the caller graph for this function:



5.9.3.8 int rmnotifynode (usbdevice * kb, int notify)

Definition at line 129 of file devnode.c.

References _rmnotifynode(), euid_guard_start, and euid_guard_stop.

Referenced by readcmd().



Here is the caller graph for this function:



5.9.3.9 void updateconnected ()

Definition at line 81 of file devnode.c.

References _updateconnected(), euid_guard_start, and euid_guard_stop.

Referenced by _setupusb(), and closeusb().

Here is the call graph for this function:



Here is the caller graph for this function:



5.9.4 Variable Documentation

5.9.4.1 const char* const devpath

Definition at line 8 of file devnode.h.

5.9.4.2 long gid

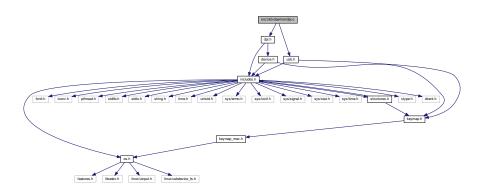
Definition at line 16 of file devnode.c.

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

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

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

Include dependency graph for dpi.c:



Functions

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

5.10.1 Function Documentation

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

Definition at line 4 of file dpi.c.

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

```
4
5   int disable = 0;
6   ushort x, y;
7   // Try to scan X,Y values
8   if(sscanf(values, "%hu,%hu", &x, &y) != 2){
```

```
// If that doesn't work, scan single number
if(sscanf(values, "%hu", &x) == 1)
10
11
            else if(!strncmp(values, "off", 3))
    // If the right side says "off", disable the level(s)
    disable = 1;
12
1.3
14
15
             else
                 // Otherwise, quit
17
                  return;
18
        if((x == 0 || y == 0) && !disable)
19
20
             return:
        // Scan the left side for stage numbers (comma-separated)
22
        int left = strlen(stages);
23
        int position = 0, field = 0;
        char stagename[3];
        while(position < left && sscanf(stages + position, "%2[^,]%n", stagename, &field) == 1){</pre>
25
             uchar stagenum;
26
             if(sscanf(stagename, "%hhu", &stagenum) && stagenum < DPI_COUNT){</pre>
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 {
                     mode->dpi.enabled |= 1 << stagenum;</pre>
35
                      mode \rightarrow dpi.x[stagenum] = x;
36
                      mode->dpi.y[stagenum] = y;
37
                 }
38
            if(stages[position += field] == ',')
39
40
                 position++;
42 }
```

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

Definition at line 44 of file dpi.c.

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

```
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 }
```

5.10.1.3 void cmd lift (usbdevice * kb, usbmode * mode, int dummy1, int dummy2, const char * height)

Definition at line 53 of file dpi.c.

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

```
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 }
```

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

Definition at line 62 of file dpi.c.

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

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

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

Definition at line 152 of file dpi.c.

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

Referenced by cmd_hwload_mouse().

```
152
          // Ask for settings
153
          uchar data_pkt[4][MSG_SIZE] = {
154
                { 0x0e, 0x13, 0x05, 1, },
155
                { 0x0e, 0x13, 0x02, 1, },
157
                { 0x0e, 0x13, 0x03, 1, },
158
                { 0x0e, 0x13, 0x04, 1, }
159
          uchar in_pkt[4][MSG_SIZE];
160
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");
return -3;
165
166
167
               }
168
169
          // Copy data from device
          dpi->enabled = in_pkt[0][4];
dpi->enabled &= (1 << DPI_COUNT) - 1;
dpi->current = in_pkt[1][4];
170
171
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
          dpi->snap = !!in_pkt[3][4];
178
179
180
          // Get X/Y DPIs
181
          for(int i = 0; i < DPI_COUNT; i++) {</pre>
182
                uchar data_pkt[MSG_SIZE] = { 0x0e, 0x13, 0xd0, 1 };
183
                uchar in_pkt[MSG_SIZE];
184
               data_pkt[2] |= i;
if(!usbrecv(kb, data_pkt, in_pkt))
185
186
                    return -2;
187
                if (memcmp(in_pkt, data_pkt, 4)) {
188
                    ckb_err("Bad input header\n");
189
                    return -3;
190
                // Copy to profile
191
               dpi->x[i] = *(ushort*)(in_pkt + 5);
dpi->y[i] = *(ushort*)(in_pkt + 7);
192
193
               light->r[LED_MOUSE + N_MOUSE_ZONES + i] = in_pkt[9];
light->g[LED_MOUSE + N_MOUSE_ZONES + i] = in_pkt[10];
light->b[LED_MOUSE + N_MOUSE_ZONES + i] = in_pkt[11];
194
195
196
197
          // Finished. Set SW DPI light to the current hardware level
198
          light->r[LED_MOUSE + 2] = light->r[LED_MOUSE +
199
        N_MOUSE_ZONES + dpi->current];
200
          light->g[LED_MOUSE + 2] = light->g[LED_MOUSE +
       N_MOUSE_ZONES + dpi->current];
light->b[LED_MOUSE + 2] = light->b[LED_MOUSE +
N_MOUSE_ZONES + dpi->current];
2.01
202
          return 0;
203 }
```



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

Definition at line 69 of file dpi.c.

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

Referenced by cmd get().

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

Here is the caller graph for this function:



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

Definition at line 124 of file dpi.c.

References lighting::b, dpiset::current, DPI_COUNT, dpiset::enabled, lighting::g, LED_MOUSE, dpiset::lift, MSG_SIZE, N_MOUSE_ZONES, lighting::r, dpiset::snap, usbsend, dpiset::y.

Referenced by cmd_hwsave_mouse().

```
124
125
          // Send X/Y DPIs
          for(int i = 0; i < DPI_COUNT; i++) {</pre>
126
               uchar data_pkt[MSG_SIZE] = { 0x07, 0x13, 0xd0, 1 };
127
128
               data_pkt[2] \mid = i;
               *(ushort*)(data_pkt + 5) = dpi->x[i];

*(ushort*)(data_pkt + 7) = dpi->y[i];
129
130
131
               \ensuremath{//} Save the RGB value for this setting too
132
               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];
133
134
               if(!usbsend(kb, data_pkt, 1))
135
136
                    return -1;
137
138
          // Send settings
139
          uchar data_pkt[4][MSG_SIZE] = {
140
               { 0x07, 0x13, 0x05, 1, dpi->enabled }, { 0x07, 0x13, 0x02, 1, dpi->current },
141
142
143
               { 0x07, 0x13, 0x03, 1, dpi->lift },
144
               { 0x07, 0x13, 0x04, 1, dpi->snap, 0x05 }
145
          if(!usbsend(kb, data_pkt[0], 4))
146
147
               return -2;
148
          // Finished
149
          return 0;
150 }
```

Here is the caller graph for this function:



5.10.1.8 int updatedpi (usbdevice * kb, int force)

Definition at line 89 of file dpi.c.

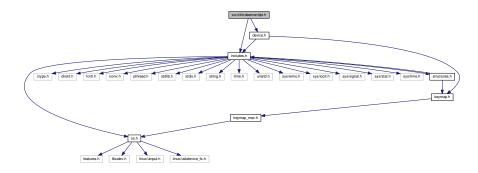
References usbdevice::active, dpiset::current, usbprofile::currentmode, usbmode::dpi, DPI_COUNT, dpiset::enabled, dpiset::forceupdate, usbprofile::lastdpi, dpiset::lift, MSG_SIZE, usbdevice::profile, dpiset::snap, usbsend, dpiset::x, and dpiset::y.

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

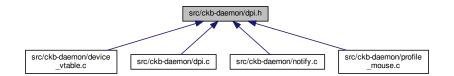
5.11 src/ckb-daemon/dpi.h File Reference

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

Include dependency graph for dpi.h:



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



Functions

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

5.11.1 Function Documentation

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

Definition at line 4 of file dpi.c.

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

```
5
       int disable = 0;
      ushort x, y;
// Try to scan X,Y values
if(sscanf(values, "%hu,%hu", &x, &y) != 2){
8
           // If that doesn't work, scan single number
if(sscanf(values, "%hu", &x) == 1)
9
10
11
                 y = x;
            else if(!strncmp(values, "off", 3))

// If the right side says "off", disable the level(s)
14
                  disable = 1;
15
            else
                 // Otherwise, quit
16
17
                  return:
18
19
        if((x == 0 || y == 0) && !disable)
20
             return;
        \ensuremath{//} Scan the left side for stage numbers (comma-separated)
2.1
22
        int left = strlen(stages);
        int position = 0, field = 0;
23
        char stagename[3];
25
        while(position < left && sscanf(stages + position, "%2[^,]%n", stagename, &field) == 1){</pre>
             if(sscanf(stagename, "%hhu", &stagenum) && stagenum < DPI_COUNT){</pre>
27
2.8
                  // Set DPI for this stage
                 if(disable){
29
30
                      mode->dpi.enabled &= ~(1 << stagenum);</pre>
                      mode->dpi.x[stagenum] = 0;
                      mode->dpi.y[stagenum] = 0;
33
                  } else {
34
                      mode->dpi.enabled |= 1 << stagenum;
                      mode \rightarrow dpi.x[stagenum] = x;
35
                      mode->dpi.y[stagenum] = y;
36
38
39
             if (stages[position += field] == ',')
40
                 position++;
41
42 }
```

5.11.1.2 void cmd dpisel (usbdevice * kb, usbmode * mode, int dummy1, int dummy2, const char * stage)

Definition at line 44 of file dpi.c.

References dpiset::current, usbmode::dpi, and DPI COUNT.

```
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 }
```

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

Definition at line 53 of file dpi.c.

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

```
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 }
```

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

Definition at line 62 of file dpi.c.

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

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

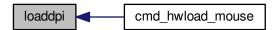
Definition at line 152 of file dpi.c.

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

Referenced by cmd_hwload_mouse().

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

Here is the caller graph for this function:



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

Definition at line 69 of file dpi.c.

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

Referenced by cmd get().

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

Here is the caller graph for this function:



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

Definition at line 124 of file dpi.c.

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

Referenced by cmd_hwsave_mouse().

```
124
125
          // Send X/Y DPIs
          for(int i = 0; i < DPI_COUNT; i++) {</pre>
126
              uchar data_pkt[MSG_SIZE] = { 0x07, 0x13, 0xd0, 1 };
127
128
               data_pkt[2] \mid = i;
              *(ushort*)(data_pkt + 5) = dpi->x[i];

*(ushort*)(data_pkt + 7) = dpi->y[i];
129
130
131
               \ensuremath{//} Save the RGB value for this setting too
132
               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];
133
134
              if(!usbsend(kb, data_pkt, 1))
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 },
141
142
143
               { 0x07, 0x13, 0x03, 1, dpi->lift },
144
               { 0x07, 0x13, 0x04, 1, dpi->snap, 0x05 }
145
          if(!usbsend(kb, data_pkt[0], 4))
146
147
               return -2;
148
          // Finished
149
          return 0;
150 }
```



5.11.1.8 int updatedpi (usbdevice * kb, int force)

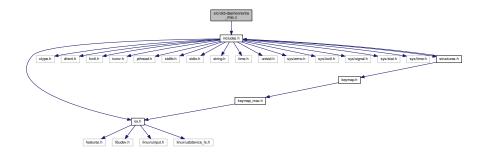
Definition at line 89 of file dpi.c.

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

```
89
       if(!kb->active)
91
            return 0;
92
       dpiset* lastdpi = &kb->profile->lastdpi;
       dpiset* newdpi = &kb->profile->currentmode->dpi;
93
       // Don't do anything if the settings haven't changed
94
       if(!force && !lastdpi->forceupdate && !newdpi->forceupdate
95
96
               && !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
102
             uchar data_pkt[MSG_SIZE] = { 0x07, 0x13, 0xd0, 0 };
103
             data_pkt[2] |= i;
104
             *(ushort*)(data_pkt + 5) = newdpi->x[i];
             *(ushort*)(data_pkt + 7) = newdpi->y[i];
105
106
             if(!usbsend(kb, data_pkt, 1))
107
                 return -1;
108
109
110
        // Send settings
        uchar data_pkt[4][MSG_SIZE] = {
111
             { 0x07, 0x13, 0x05, 0, newdpi->enabled }, { 0x07, 0x13, 0x02, 0, newdpi->current },
112
113
114
             { 0x07, 0x13, 0x03, 0, newdpi->lift },
```

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

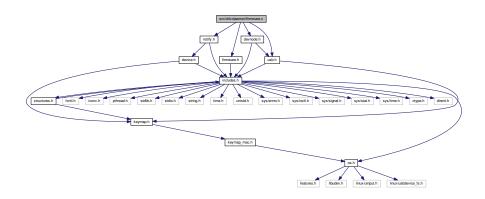
```
#include "includes.h"
Include dependency graph for extra_mac.c:
```



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

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

Include dependency graph for firmware.c:



Macros

- #define FW_OK 0
- #define FW_NOFILE -1
- #define FW WRONGDEV -2
- #define FW_USBFAIL -3
- #define FW_MAXSIZE (255 * 256)

Functions

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

5.13.1 Macro Definition Documentation

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

Definition at line 51 of file firmware.c.

Referenced by fwupdate().

```
5.13.1.2 #define FW_NOFILE -1
```

Definition at line 7 of file firmware.c.

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

```
5.13.1.3 #define FW_OK 0
```

Definition at line 6 of file firmware.c.

Referenced by cmd_fwupdate(), and fwupdate().

```
5.13.1.4 #define FW_USBFAIL -3
```

Definition at line 9 of file firmware.c.

Referenced by cmd_fwupdate(), and fwupdate().

```
5.13.1.5 #define FW_WRONGDEV -2
```

Definition at line 8 of file firmware.c.

Referenced by cmd_fwupdate(), and fwupdate().

5.13.2 Function Documentation

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

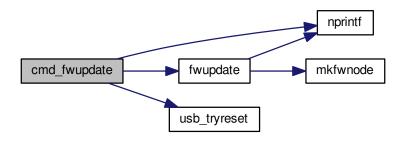
Definition at line 154 of file firmware.c.

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

```
if(!HAS_FEATURES(kb, FEAT_FWUPDATE))
156
             return 0;
        // Update the firmware
157
        int ret = fwupdate(kb, path, nnumber);
while(ret == FW_USBFAIL){
158
159
               Try to reset the device if it fails
160
161
            if(usb_tryreset(kb))
162
163
            ret = fwupdate(kb, path, nnumber);
164
165
        switch(ret){
166
        case FW_OK:
167
            nprintf(kb, nnumber, 0, "fwupdate %s ok\n", path);
```

```
168
           break;
169
        case FW_NOFILE:
170
        case FW_WRONGDEV:
           nprintf(kb, nnumber, 0, "fwupdate %s invalid\n", path);
171
172
            break:
173
        case FW_USBFAIL:
174
           nprintf(kb, nnumber, 0, "fwupdate %s fail\n", path);
175
176
177
        return 0;
178 }
```

Here is the call graph for this function:



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

Definition at line 55 of file firmware.c.

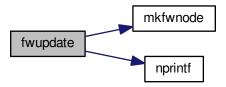
References ckb_err, ckb_info, FW_MAXSIZE, FW_NOFILE, FW_OK, FW_USBFAIL, FW_WRONGDEV, usbdevice::fwversion, mkfwnode(), MSG_SIZE, nprintf(), usbdevice::product, usbdevice::usbdelay, usbsend, and usbdevice::vendor.

Referenced by cmd_fwupdate().

```
56
        \ensuremath{//} Read the firmware from the given path
57
        char* fwdata = calloc(1, FW MAXSIZE + 256);
58
        int fd = open(path, O_RDONLY);
59
        if(fd == -1){
            ckb_err("Failed to open firmware file %s: %s\n", path, strerror(errno));
             return FW_NOFILE;
62
        ssize_t length = read(fd, fwdata, FW_MAXSIZE + 1);
63
        ckb_err("Failed to read firmware file %s: %s\n", path, length <= 0 ? strerror(errno) : "
64
65
       Wrong size");
66
            close(fd);
67
             return FW_NOFILE;
68
        close(fd);
69
70
71
        short vendor, product, version;
        // Copy the vendor ID, product ID, and version from the firmware file
73
        memcpy(&vendor, fwdata + 0x102, 2);
74
        memcpy(&product, fwdata + 0x104, 2);
        memcpy(&version, fwdata + 0x106, 2);
// Check against the actual device
75
76
        if(vendor != kb->vendor || product != kb->product) {
78
            ckb_err("Firmware file %s doesn't match device (V: %04x P: %04x)\n", path, vendor, product);
79
             return FW_WRONGDEV;
80
        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)
```

```
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 },
{ 0x7f, 0x01, 0x3c, 0 },
87
88
89
            { 0x7f, 0x02, 0x3c, 0 },
90
91
            { 0x7f, 0x03, 0x3c, 0 },
92
             { 0x7f, 0x04, 0x3c, 0 },
93
            \{ 0x7f, 0x05, 0x10, 0 \}
94
       };
       int output = 0, last = 0;
95
       int index = 0;
while(output < length) {</pre>
96
98
            int npackets = 1;
            // Packet 1: data position
data_pkt[1][6] = index++;
while(output < length){</pre>
99
100
101
                 npackets++;
102
103
                  if(npackets != 6) {
104
                       // Packets 2-5: 60 bytes of data
105
                       memcpy(data_pkt[npackets] + 4, fwdata + output, 60);
106
                       last = output;
107
                       output += 60;
                  } else {
// Packet 6: 16 bytes
108
109
110
                       memcpy(data_pkt[npackets] + 4, fwdata + output, 16);
111
                       last = output;
112
                       output += 16;
113
                       break;
114
                  }
115
116
              if(index == 1){
117
                  if(!usbsend(kb, data_pkt[0], 1)){
118
                       ckb_err("Firmware update failed\n");
                       return FW_USBFAIL;
119
120
                  // The above packet can take a lot longer to process, so wait for a while
121
122
                  sleep(3);
123
                  if(!usbsend(kb, data_pkt[2], npackets - 1)){
124
                       ckb_err("Firmware update failed\n");
                       return FW_USBFAIL;
125
126
127
             } else {
128
                  // If the output ends here, set the length byte appropriately
129
                  if(output >= length)
                       data_pkt[npackets][2] = length - last;
130
                  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
         ^{\prime}// Send the final pair of messages
138
         uchar data_pkt2[2][MSG_SIZE] = {
139
             { 0x07, 0x0d, 0xf0, 0x00, 0x00, 0x00, index }, { 0x07, 0x02, 0xf0, 0 }
140
141
142
         if(!usbsend(kb, data_pkt2[0], 2)){
143
             ckb_err("Firmware update failed\n");
144
             return FW_USBFAIL;
145
146
147
         // Updated successfully
148
         kb->fwversion = version;
149
         mkfwnode(kb);
150
         ckb_info("Firmware update complete\n");
         return FW_OK;
151
152 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.13.2.3 int getfwversion (usbdevice * kb)

Definition at line 11 of file firmware.c.

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

Referenced by _start_dev().

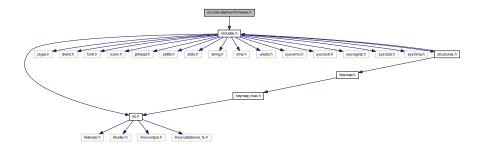
```
11
12
        // Ask board for firmware info
       uchar data_pkt[MSG_SIZE] = { 0x0e, 0x01, 0 };
13
14
        uchar in_pkt[MSG_SIZE];
        if(!usbrecv(kb, data_pkt, in_pkt))
15
        if(in_pkt[0] != 0x0e || in_pkt[1] != 0x01){
    ckb_err("Bad input header\n");
17
18
            return -1;
19
20
21
        short vendor, product, version, bootloader;
22
        // Copy the vendor ID, product ID, version, and poll rate from the firmware data
2.3
        memcpy(&version, in_pkt + 8, 2);
       memcpy(&bootloader, in_pkt + 10, 2);
memcpy(&vendor, in_pkt + 12, 2);
memcpy(&product, in_pkt + 14, 2);
2.4
25
26
       char poll = in_pkt[16];
28
        if (poll <= 0) {
29
            poll = -1;
            kb->features &= ~FEAT_POLLRATE;
30
31
       // Print a warning if the message didn't match the expected data
32
        if (vendor != kb->vendor)
33
            ckb_warn("Got vendor ID %04x (expected %04x)\n", vendor, kb->
       vendor);
35
        if(product != kb->product)
            ckb_warn("Got product ID 04x (expected 404x)\n", product, kb->
36
       product);
37
        // Set firmware version and poll rate
38
        if(version == 0 || bootloader == 0){
```

```
39
        // Needs firmware update
40
        kb->fwversion = 0;
        kb \rightarrow pollrate = -1;
41
42
     } else {
        4.3
44
    fwversion);
45
        kb->fwversion = version;
46
        kb->pollrate = poll;
47
48
     return 0;
49 }
```

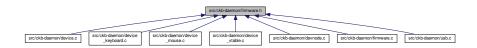


5.14 src/ckb-daemon/firmware.h File Reference

```
#include "includes.h"
Include dependency graph for firmware.h:
```



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



Functions

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

5.14.1 Function Documentation

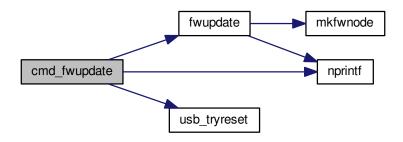
5.14.1.1 int cmd_fwupdate (usbdevice * kb, usbmode * dummy1, int nnumber, int dummy2, const char * path)

Definition at line 154 of file firmware.c.

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

```
154
155
        if(!HAS_FEATURES(kb, FEAT_FWUPDATE))
156
            return 0;
        // Update the firmware
158
        int ret = fwupdate(kb, path, nnumber);
159
        while(ret == FW_USBFAIL) {
160
            // Try to reset the device if it fails
            if(usb_tryreset(kb))
161
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
            break;
        case FW_USBFAIL:
173
174
           nprintf(kb, nnumber, 0, "fwupdate %s fail\n", path);
175
            return -1;
177
        return 0;
178 }
```

Here is the call graph for this function:



5.14.1.2 int getfwversion (usbdevice * kb)

Definition at line 11 of file firmware.c.

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

Referenced by _start_dev().

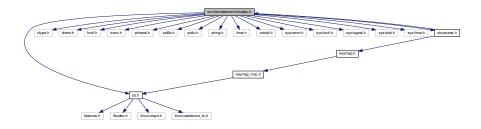
```
20
        , short vendor, product, version, bootloader; // Copy the vendor ID, product ID, version, and poll rate from the firmware data
21
22
        memcpy(&version, in_pkt + 8, 2);
2.3
        memcpy(&bootloader, in_pkt + 10, 2);
memcpy(&vendor, in_pkt + 12, 2);
memcpy(&product, in_pkt + 14, 2);
2.4
25
26
27
        char poll = in_pkt[16];
        if (pol1 <= 0) {
   pol1 = -1;
28
29
30
            kb->features &= ~FEAT_POLLRATE;
31
        // Print a warning if the message didn't match the expected data
32
33
       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
        // Set firmware version and poll rate
38
        if(version == 0 || bootloader == 0){
39
             // Needs firmware update
40
             kb->fwversion = 0;
            kb->pollrate = -1;
41
42
        } else {
            if(version != kb->fwversion && kb->fwversion != 0)
43
44
                 ckb_warn("Got firmware version 04x (expected 04x)\n", version, kb->
       fwversion);
4.5
            kb->fwversion = version;
46
            kb->pollrate = poll;
47
48
        return 0;
49 }
```



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

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

Include dependency graph for includes.h:



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



Macros

- #define INDEX OF(entry, array) (int)(entry array)
- #define ckb_s_out stdout
- #define ckb s err stdout
- #define __FILE_NOPATH__ (strrchr(__FILE__, '/') ? strrchr(__FILE__, '/') + 1 : __FILE__)
- #define ckb fatal nofile(fmt, args...) fprintf(ckb s err, "[F] " fmt, ## args)
- #define ckb_fatal_fn(fmt, file, line, args...) fprintf(ckb_s_err, "[F] %s (via %s:%d): "fmt, __func__, file, line, ## args)
- #define ckb_fatal(fmt, args...) fprintf(ckb_s_err, "[F] %s (%s:%d): " fmt, __func__, __FILE_NOPATH__, __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)

5.15.1 Macro Definition Documentation

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

Definition at line 40 of file includes.h.

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

Definition at line 49 of file includes.h.

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

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

Definition at line 48 of file includes.h.

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

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

Definition at line 47 of file includes.h.

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

Definition at line 46 of file includes.h.

Referenced by usbmain().

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

Definition at line 45 of file includes.h.

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

Definition at line 44 of file includes.h.

Referenced by main().

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

Definition at line 55 of file includes.h.

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

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

Definition at line 54 of file includes.h.

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

Definition at line 53 of file includes.h.

Referenced by main().

5.15.1.11 #define ckb_s_err stdout

Definition at line 36 of file includes.h.

5.15.1.12 #define ckb_s_out stdout

Definition at line 35 of file includes.h.

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

Definition at line 52 of file includes.h.

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

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

Definition at line 51 of file includes.h.

Referenced by os_usbrecv(), and os_usbsend().

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

Definition at line 50 of file includes.h.

Referenced by main().

5.15.1.16 #define INDEX_OF(entry, array) (int)(entry - array)

Definition at line 27 of file includes.h.

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

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

Definition at line 60 of file includes.h.

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

Definition at line 61 of file includes.h.

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

Definition at line 59 of file includes.h.

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

Definition at line 63 of file includes.h.

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

Definition at line 62 of file includes.h.

5.15.2 Typedef Documentation

5.15.2.1 typedef unsigned char uchar

Definition at line 24 of file includes.h.

5.15.2.2 typedef unsigned short ushort

Definition at line 25 of file includes.h.

5.15.3 Function Documentation

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

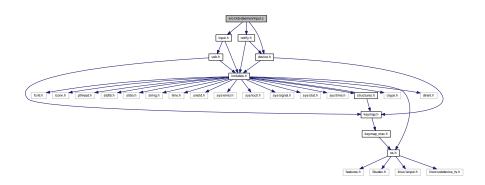
Definition at line 19 of file main.c.

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

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

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

Include dependency graph for input.c:



Macros

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

Functions

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

5.16.1 Macro Definition Documentation

5.16.1.1 #define IS_WHEEL(scan, kb) (((scan) == KEY_VOLUMEUP || (scan) == KEY_VOLUMEDOWN || (scan) == BTN WHEELDOWN) && !IS K65(kb))

Referenced by inputupdate keys().

5.16.2 Function Documentation

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

Definition at line 226 of file input.c.

References keymacro::actioncount, keymacro::actions, usbmode::bind, keymacro::combo, macroaction::down, keymap, MACRO_MAX, binding::macrocap, binding::macrocount, binding::macros, N_KEYBYTES_INPUT, N_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;
             252
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)
265
266
             return:
267
         // Count the number of actions (comma separated)
         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
         \ensuremath{//} 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 }
```

Here is the caller graph for this function:



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

Definition at line 188 of file input.c.

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

```
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 <
192
193
      N_KEYS_INPUT) {
194
            pthread_mutex_lock(imutex(kb));
195
             mode->bind.base[keyindex] = tocode;
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 }
```

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

Definition at line 332 of file input.c.

References _cmd_macro(), and imutex.

```
332

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

Here is the call graph for this function:



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

Definition at line 218 of file input.c.

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

```
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 }
```

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

Definition at line 210 of file input.c.

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

```
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 }
```

5.16.2.6 void freebind (binding * bind)

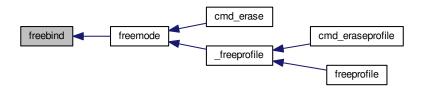
Definition at line 181 of file input.c.

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

Referenced by freemode().

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

Here is the caller graph for this function:



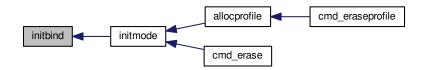
5.16.2.7 void initbind (binding * bind)

Definition at line 173 of file input.c.

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

Referenced by initmode().

Here is the caller graph for this function:



5.16.2.8 void inputupdate (usbdevice *kb)

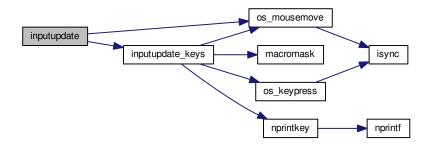
Definition at line 122 of file input.c.

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

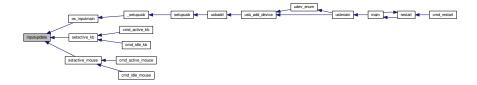
Referenced by os_inputmain(), setactive_kb(), and setactive_mouse().

```
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
            // Finish up
138
139
           memcpy(input->prevkeys, input->keys, N_KEYBYTES_INPUT);
140 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



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

Definition at line 15 of file input.c.

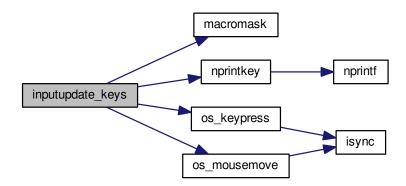
References keymacro::actioncount, keymacro::actions, usbdevice::active, binding::base, usbmode::bind, keymacro::combo, usbprofile::currentmode, usbdevice::delay, macroaction::down, usbdevice::input, IS_MOD, IS_WHEEL, keymap, usbinput::keys, binding::macrocount, macromask(), binding::macros, N_KEYBYTES_INPUT, N_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
18
       usbinput * input = &kb->input;
       // 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) {
29
                           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
50
        // Make a list of keycodes to send. Rearrange them so that modifier keydowns always come first
51
        // and modifier keyups always come last. This ensures that shortcut keys will register properly
52
        // even if both keydown events happen at once.
53
        // N_KEYS + 4 is used because the volume wheel generates keydowns and keyups at the same time
        // (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
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:



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

Definition at line 5 of file input.c.

References N_KEYBYTES_INPUT.

Referenced by inputupdate_keys().

Here is the caller graph for this function:



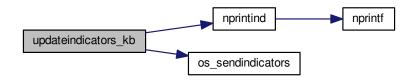
5.16.2.11 void updateindicators_kb (usbdevice*kb, int force)

Definition at line 142 of file input.c.

References usbdevice::active, usbprofile::currentmode, DELAY_SHORT, usbdevice::hw_ileds, usbdevice::hw_ileds_old, I_CAPS, I_NUM, I_SCROLL, usbdevice::ileds, usbmode::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;
uchar indicators[] = { I_NUM, I_CAPS, I_SCROLL };
160
161
          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++) {
   if (mode->inotify[notify] & mask)
166
167
                         nprintind(kb, notify, mask, hw_new & mask);
169
170
          }
171 }
```

Here is the call graph for this function:

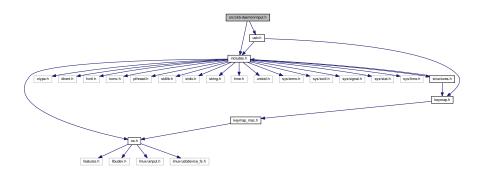


5.17 src/ckb-daemon/input.h File Reference

#include "includes.h"

#include "usb.h"

Include dependency graph for input.h:



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



Macros

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

Functions

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

5.17.1 Macro Definition Documentation

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

Definition at line 34 of file input.h.

Referenced by inputupdate_keys().

5.17.2 Function Documentation

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

Definition at line 188 of file input.c.

References binding::base, usbmode::bind, imutex, keymap, N KEYS INPUT, and key::scan.

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

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

Definition at line 332 of file input.c.

References _cmd_macro(), and imutex.

Here is the call graph for this function:



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

Definition at line 218 of file input.c.

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

```
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 }
```

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

Definition at line 210 of file input.c.

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

```
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 }
```

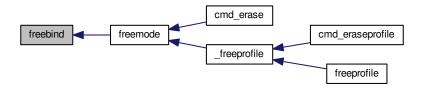
5.17.2.5 void freebind (binding * bind)

Definition at line 181 of file input.c.

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

Referenced by freemode().

Here is the caller graph for this function:



5.17.2.6 void initbind (binding * bind)

Definition at line 173 of file input.c.

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

Referenced by initmode().

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

Here is the caller graph for this function:



5.17.2.7 void inputupdate (usbdevice * kb)

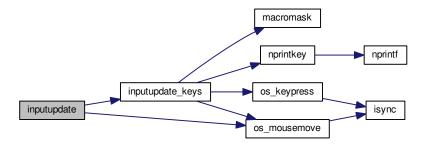
Definition at line 122 of file input.c.

References usbdevice::input, inputupdate_keys(), os_mousemove(), usbdevice::profile, usbinput::rel_x, usbinput::rel_y, usbdevice::uinput kb, and usbdevice::uinput mouse.

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

```
123 #ifdef OS_LINUX
124
          if((!kb->uinput_kb || !kb->uinput_mouse)
124 - ...
125 #else
126 if(!kb->event
127 #endif
                    || !kb->profile)
128
129
               return;
130
          // Process key/button input
131
          inputupdate_keys(kb);
          // Process mouse movement
usbinput* input = &kb->input;
if(input->rel_x != 0 || input->rel_y != 0){
132
133
134
               os_mousemove(kb, input->rel_x, input->rel_y);
input->rel_x = input->rel_y = 0;
135
136
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:



5.17.2.8 void os_inputclose (usbdevice * kb)

Definition at line 76 of file input_linux.c.

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

Referenced by closeusb().

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

Here is the caller graph for this function:



5.17.2.9 int os_inputopen (usbdevice * kb)

Parameters

```
kb
```

Returns

Some tips on using uinput_user_dev in

Definition at line 55 of file input_linux.c.

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

Referenced by _setupusb().

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

Here is the call graph for this function:



Here is the caller graph for this function:



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

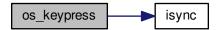
Definition at line 118 of file input linux.c.

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

Referenced by inputupdate_keys().

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

Here is the call graph for this function:



Here is the caller graph for this function:



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

Definition at line 143 of file input_linux.c.

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

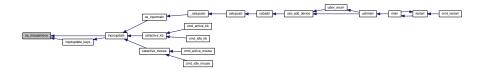
Referenced by inputupdate(), and inputupdate_keys().

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

Here is the call graph for this function:



Here is the caller graph for this function:



5.17.2.12 int os_setupindicators (usbdevice * kb)

Definition at line 189 of file input_linux.c.

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

Referenced by _setupusb().

```
190
         // Initialize LEDs to all off
        kb->hw_ileds = kb->hw_ileds_old = kb->ileds = 0;
191
        \ensuremath{//} Create and detach thread to read LED events
192
193
        pthread_t thread;
        int err = pthread_create(&thread, 0, _ledthread, kb);
if(err != 0)
194
195
196
             return err;
197
        pthread_detach(thread);
198
         return 0;
199 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



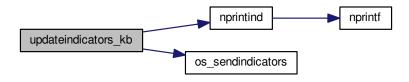
5.17.2.13 void updateindicators_kb (usbdevice * kb, int force)

Definition at line 142 of file input.c.

References usbdevice::active, usbprofile::currentmode, DELAY_SHORT, usbdevice::hw_ileds, usbdevice::hw_ileds_old, I_CAPS, I_NUM, I_SCROLL, usbdevice::ileds, usbmode::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
147
         if (kb->active) {
148
             usbmode* mode = kb->profile->currentmode;
149
             new = (new & ~mode->ioff) | mode->ion;
150
         kb->ileds = new;
151
         kb->hw_ileds_old = hw_new;
152
153
         if(old != new || force) {
154
             DELAY_SHORT(kb);
155
             os_sendindicators(kb);
156
         // Print notifications if desired
157
         if(!kb->active)
158
159
             return;
160
         usbmode* mode = kb->profile->currentmode;
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
             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 }
```

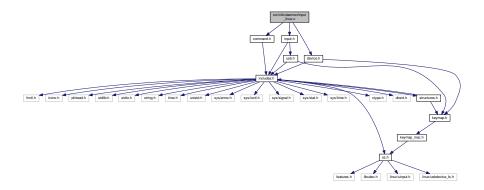
Here is the call graph for this function:



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

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

Include dependency graph for input_linux.c:



Functions

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

os_inputopen

- void os_inputclose (usbdevice *kb)
- static void isync (usbdevice *kb)
- void os_keypress (usbdevice *kb, int scancode, int down)
- void os_mousemove (usbdevice *kb, int x, int y)
- void * _ledthread (void *ctx)
- int os_setupindicators (usbdevice *kb)

5.18.1 Function Documentation

```
5.18.1.1 void* _ledthread ( void * ctx )
```

Definition at line 165 of file input_linux.c.

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

Referenced by os_setupindicators().

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

Here is the caller graph for this function:



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

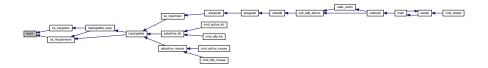
Definition at line 107 of file input_linux.c.

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

Referenced by os_keypress(), and os_mousemove().

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

Here is the caller graph for this function:



5.18.1.3 void os_inputclose (usbdevice * kb)

Definition at line 76 of file input_linux.c.

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

Referenced by closeusb().

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

```
100    // Close the mouse
101    ioctl(kb->uinput_mouse - 1, UI_DEV_DESTROY);
102    close(kb->uinput_mouse - 1);
103    kb->uinput_mouse = 0;
104 )
```

Here is the caller graph for this function:



5.18.1.4 int os_inputopen (usbdevice * kb)

Parameters

```
kb |
```

Returns

Some tips on using uinput_user_dev in

Definition at line 55 of file input linux.c.

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

Referenced by _setupusb().

```
55
         // Create the new input device
57
         int index = INDEX_OF(kb, keyboard);
58
         struct uinput_user_dev indev;
        memset(&indev, 0, sizeof(indev));
snprintf(indev.name, UINPUT_MAX_NAME_SIZE, "ckb%d: %s", index, kb->name);
59
60
         indev.id.bustype = BUS_USB;
         indev.id.vendor = kb->vendor;
62
        indev.id.product = kb->product;
indev.id.version = kb->fwversion;
64
6.5
         // Open keyboard
        int fd = uinputopen(&indev, 0);
kb->uinput_kb = fd;
if(fd <= 0)</pre>
66
67
69
              return 0;
         // Open mouse
70
         fd = uinputopen(&indev, 1);
71
         kb->uinput_mouse = fd;
return fd <= 0;</pre>
72
73
```

Here is the call graph for this function:



Here is the caller graph for this function:



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

Definition at line 118 of file input_linux.c.

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

Referenced by inputupdate keys().

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

Here is the call graph for this function:



Here is the caller graph for this function:



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

Definition at line 143 of file input_linux.c.

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

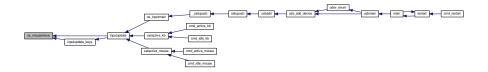
Referenced by inputupdate(), and inputupdate keys().

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

Here is the call graph for this function:



Here is the caller graph for this function:



5.18.1.7 int os_setupindicators (usbdevice * kb)

Definition at line 189 of file input_linux.c.

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

Referenced by _setupusb().

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

Here is the call graph for this function:



Here is the caller graph for this function:



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

Definition at line 9 of file input_linux.c.

References ckb_err, and ckb_warn.

Referenced by os_inputopen().

```
10
         int fd = open("/dev/uinput", O_RDWR);
11
          if(fd < 0){
               // If that didn't work, try /dev/input/uinput instead
fd = open("/dev/input/uinput", O_RDWR);
if(fd < 0){</pre>
12
13
14
                    ckb_err("Failed to open uinput: %s\n", strerror(errno));
15
                     return 0;
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
21
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
28
         } 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
35
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));
40
         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;
```

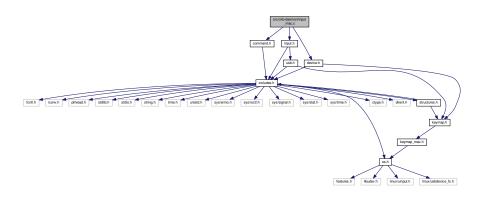
Here is the caller graph for this function:



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

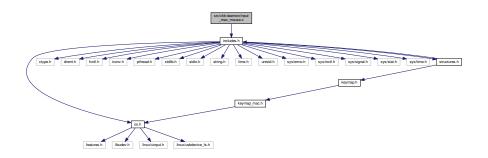
```
#include "command.h"
#include "device.h"
#include "input.h"
```

Include dependency graph for input_mac.c:



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

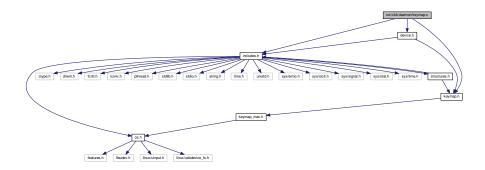
#include "includes.h"
Include dependency graph for input_mac_mouse.c:



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

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

Include dependency graph for keymap.c:



Macros

• #define BUTTON_HID_COUNT 5

Functions

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

Variables

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

5.21.1 Macro Definition Documentation

5.21.1.1 #define BUTTON_HID_COUNT 5

Definition at line 364 of file keymap.c.

Referenced by corsair_mousecopy(), and hid_mouse_translate().

5.21.2 Function Documentation

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

Definition at line 394 of file keymap.c.

References N KEYBYTES HW.

Referenced by os_inputmain().

Here is the caller graph for this function:



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

Definition at line 403 of file keymap.c.

References BUTTON_HID_COUNT, CLEAR_KEYBIT, MOUSE_BUTTON_FIRST, N_BUTTONS_HW, and SET_K-EYBIT.

Referenced by os_inputmain().

```
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;
            uchar test = 1 << (bit % 8);
411
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:



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

Definition at line 223 of file keymap.c.

References ckb_warn, CLEAR_KEYBIT, and SET_KEYBIT.

Referenced by os_inputmain().

```
223
224
        if(length < 1)</pre>
225
        // LUT for HID \rightarrow 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]
                                              52,
229
                                                    39,
                                                         27,
             -1, -1, -1,
                             -1,
                                        54,
                                                               40,
230
                       33,
                                   25,
                                        28,
                                              38,
                                                   29,
                                                              53,
                                                                    26,
                                                                                     50,
                                   19,
231
                  16,
                        17,
                             18,
                                        20,
                                                         82,
                                                               Ο,
                                                                    86,
                                                                               64,
                                                                                     23,
                                                                                                35,
232
             79.
                  80,
                       81,
                             46.
                                   47,
                                        12,
                                              57.
                                                    58,
                                                         59,
                                                              36,
                                                                     1.
                                                                                      4.
                         9,
                                        72,
                                              73,
                                                         75,
                                                                         78,
                                                                               87,
                                                                                     88,
233
              7.
                   8.
                             10.
                                   11.
                                                    74.
                                                              76.
                                                                                          89.
                       92, 102, 103, 104, 105, 106,
             93,
                  94,
                                                                                         114, 108,
234
                                                        107, 115, 116, 117, 112, 113,
235
            109, 110, 118,
                            119,
                                        69,
                                   49,
                                              -2,
                                                    -2,
                                                                    -2,
                                                                                     -2,
236
                             -2,
237
            130, 131,
                                        -2,
                                                         -2,
238
             -2,
                  -2,
                       -2,
                             -2,
                                  -2,
                                        -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.
                                                                               -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,
242
              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)
289
                                     SET_KEYBIT(kbinput, hid_codes[keybit]);
290
                                 else
                            291
292
293
                                 CLEAR_KEYBIT(kbinput, hid_codes[keybit]);
294
                       }
295
296
                  break:
              } else if(urbinput[0] == 2)
297
                          // Type 2: media key (fall through)
298
                 ;
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;
324
                   case 226:
325
                       SET_KEYBIT(kbinput, 97);
                                                         // mute
326
                       break;
327
                   case 233:
328
                       SET_KEYBIT(kbinput, 130);
                                                        // volup
```

```
break;
330
                 case 234:
331
                     SET_KEYBIT(kbinput, 131); // voldn
332
                     break;
333
                 }
334
             }
335
            break;
336
        case 3:
            // EP 3 non-RGB: NKRO input
337
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
             for(int byte = 0; byte < 14; byte++){</pre>
346
                 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];
                     if((input >> bit) & 1) {
   if(scan >= 0)
351
352
353
                             SET_KEYBIT(kbinput, hid_codes[keybit]);
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;
361
362 }
```

Here is the caller graph for this function:



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

Definition at line 366 of file keymap.c.

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

Referenced by os_inputmain().

```
366
367
         if((endpoint != 2 && endpoint != -2) || length < 10)</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];
         if(wheel > 0)
384
385
             SET_KEYBIT(kbinput, MOUSE_EXTRA_FIRST);
                                                                     // wheelup
386
         else
387
              CLEAR_KEYBIT(kbinput, MOUSE_EXTRA_FIRST);
388
         if(wheel < 0)
```

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

Here is the caller graph for this function:



5.21.3 Variable Documentation

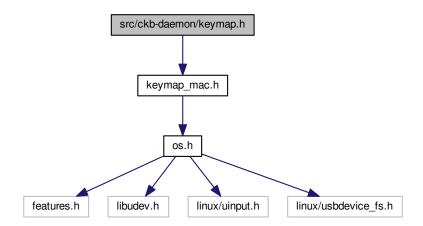
5.21.3.1 const key keymap[(((152+3+12)+25)+11)]

Definition at line 5 of file keymap.c.

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

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

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



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



Data Structures

· struct key

Macros

- #define KEY NONE -1
- #define KEY_CORSAIR -2
- #define KEY UNBOUND -3
- #define BTN WHEELUP 0x1f01
- #define BTN WHEELDOWN 0x1f02
- #define KEY_BACKSLASH_ISO KEY_BACKSLASH
- #define N KEYS HW 152
- #define N_KEYBYTES_HW ((N_KEYS_HW + 7) / 8)
- #define N_KEY_ZONES 3
- #define N_KEYS_EXTRA 12
- #define N BUTTONS HW 20
- #define N_BUTTONS_EXTENDED 25
- #define MOUSE_BUTTON_FIRST (N_KEYS_HW + N_KEY_ZONES + N_KEYS_EXTRA)
- #define MOUSE_EXTRA_FIRST (MOUSE_BUTTON_FIRST + N_BUTTONS_HW)
- #define N_KEYS_INPUT (MOUSE_BUTTON_FIRST + N_BUTTONS_EXTENDED)
- #define N_KEYBYTES_INPUT ((N_KEYS_INPUT + 7) / 8)
- #define LED MOUSE N KEYS HW
- #define N_MOUSE_ZONES 5
- #define N_MOUSE_ZONES_EXTENDED 11
- #define LED_DPI (LED_MOUSE + 2)
- #define N KEYS EXTENDED (N KEYS INPUT + N MOUSE ZONES EXTENDED)
- #define N_KEYBYTES_EXTENDED ((N_KEYS_EXTENDED + 7) / 8)
- #define SCAN_SILENT 0x8000
- #define SCAN KBD 0
- #define SCAN_MOUSE 0x1000

Functions

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

Variables

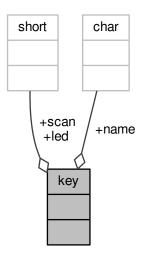
const key keymap [(((152+3+12)+25)+11)]

5.22.1 Data Structure Documentation

5.22.1.1 struct key

Definition at line 49 of file keymap.h.

Collaboration diagram for key:



Data Fields

short	led	
const char *	name	
short	scan	

5.22.2 Macro Definition Documentation

5.22.2.1 #define BTN_WHEELDOWN 0x1f02

Definition at line 13 of file keymap.h.

Referenced by os_keypress().

5.22.2.2 #define BTN_WHEELUP 0x1f01

Definition at line 12 of file keymap.h.

Referenced by os_keypress().

5.22.2.3 #define KEY_BACKSLASH_ISO KEY_BACKSLASH

Definition at line 20 of file keymap.h.

5.22.2.4 #define KEY_CORSAIR -2

Definition at line 8 of file keymap.h.

5.22.2.5 #define KEY_NONE -1

Definition at line 7 of file keymap.h.

5.22.2.6 #define KEY_UNBOUND -3

Definition at line 9 of file keymap.h.

Referenced by cmd_unbind().

5.22.2.7 #define LED_DPI (LED_MOUSE + 2)

Definition at line 43 of file keymap.h.

Referenced by loadrgb_mouse(), and savergb_mouse().

5.22.2.8 #define LED_MOUSE N_KEYS_HW

Definition at line 39 of file keymap.h.

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

5.22.2.9 #define MOUSE_BUTTON_FIRST (N_KEYS_HW + N_KEY_ZONES + N_KEYS_EXTRA)

Definition at line 33 of file keymap.h.

Referenced by corsair_mousecopy(), and hid_mouse_translate().

5.22.2.10 #define MOUSE_EXTRA_FIRST (MOUSE_BUTTON_FIRST + N_BUTTONS_HW)

Definition at line 34 of file keymap.h.

Referenced by hid_mouse_translate().

5.22.2.11 #define N_BUTTONS_EXTENDED 25

Definition at line 32 of file keymap.h.

5.22.2.12 #define N_BUTTONS_HW 20

Definition at line 31 of file keymap.h.

Referenced by corsair_mousecopy().

5.22.2.13 #define N_KEY_ZONES 3

Definition at line 27 of file keymap.h.

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

Definition at line 46 of file keymap.h.

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

Definition at line 25 of file keymap.h.

Referenced by corsair_kbcopy().

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

Definition at line 37 of file keymap.h.

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

5.22.2.17 #define N_KEYS_EXTENDED (N_KEYS_INPUT + N_MOUSE_ZONES_EXTENDED)

Definition at line 45 of file keymap.h.

Referenced by printrgb(), and readcmd().

5.22.2.18 #define N_KEYS_EXTRA 12

Definition at line 29 of file keymap.h.

5.22.2.19 #define N_KEYS_HW 152

Definition at line 24 of file keymap.h.

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

5.22.2.20 #define N_KEYS_INPUT (MOUSE_BUTTON_FIRST + N_BUTTONS_EXTENDED)

Definition at line 36 of file keymap.h.

 $Referenced \ by \ _cmd_get(), \ _cmd_macro(), \ cmd_bind(), \ cmd_notify(), \ cmd_rebind(), \ cmd_unbind(), \ initbind(), \ and \ input update \ _keys().$

5.22.2.21 #define N_MOUSE_ZONES 5

Definition at line 40 of file keymap.h.

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

5.22.2.22 #define N_MOUSE_ZONES_EXTENDED 11

Definition at line 41 of file keymap.h.

5.22.2.23 #define SCAN_KBD 0

Definition at line 57 of file keymap.h.

5.22.2.24 #define SCAN_MOUSE 0x1000

Definition at line 58 of file keymap.h.

Referenced by os_keypress().

5.22.2.25 #define SCAN_SILENT 0x8000

Definition at line 56 of file keymap.h.

Referenced by inputupdate_keys().

5.22.3 Function Documentation

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

Definition at line 394 of file keymap.c.

References N_KEYBYTES_HW.

Referenced by os_inputmain().

Here is the caller graph for this function:

```
cosair Bloopy co_irputmain __setupusb __setupusb __usbadd __usb_add_device __usb_main __main __restart __cmd_restart
```

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

Definition at line 403 of file keymap.c.

References BUTTON_HID_COUNT, CLEAR_KEYBIT, MOUSE_BUTTON_FIRST, N_BUTTONS_HW, and SET_K-EYBIT.

Referenced by os_inputmain().

```
403
404
          if (endpoint == 2 || endpoint == -2) {
405
               if(urbinput[0] != 3)
406
                    return:
407
              urbinput++;
408
409
          for(int bit = BUTTON_HID_COUNT; bit < N_BUTTONS_HW; bit++) {</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:



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

Definition at line 223 of file keymap.c.

References ckb warn, CLEAR KEYBIT, and SET KEYBIT.

Referenced by os inputmain().

```
223
224
         if(length < 1)</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,
34,
              -1,
                   -1,
                         -1,
                                     37,
                                          54,
                                                52,
                                                      39,
                                                            27,
                                                                 40,
                                                                       41.
                                                                             42,
                                                                                   32,
                                                                                         43,
                                                                                              44, 45,
                   55, 33,
                                                           31,
                                                                       26,
                                                                             51,
230
              56,
                                     25,
                                          28,
                                                38,
                                                      29,
                                                                                   30,
                                                                                         50,
                                                                                              13,
                                                                 53,
                                                           82,
                                                                                              84,
231
              15.
                   16,
                         17,
                              18,
                                    19,
                                          20,
                                                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.
                          9,
                                                            75,
                                                                             78,
                    8,
233
              7.
                               10,
                                     11,
                                          72,
                                                73,
                                                      74.
                                                                  76,
                                                                                   87,
                                                                                         88.
                                                                                               89.
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,
                                                            98,
                                    -1,
                                                                       -2,
                                                                                        -1,
237
            130, 131,
                         -1,
                               -1,
                                          -2,
                                                -1,
                                                      -2,
                                                            -2,
                                                                 -2,
                                                                                   -2,
                                                                                              -1,
                                                                                                    -1.
                              -2,
                                          -1,
                                                -1,
                                                     -1,
                                                           -1,
                                                                 -1,
                                                                             -1,
                                                                                  -1,
                                                                                        -1,
                                                                                                    -1,
                                                                       -1,
238
             -2, -2,
                        -2,
                                    -2,
                                                                                              -1,
239
             -1, -1,
                        -1,
                              -1.
                                    -1.
                                          -1,
                                                -1.
                                                     -1,
                                                           -1.
                                                                 -1,
                                                                       -1,
                                                                             -1.
                                                                                   -1.
                                                                                        -1,
                                                                                              -1.
                                                                                                    -1.
             -1.
240
                               -1.
                                    -1.
                                          -1.
                                                -1.
                                                                                   -1.
                                                                                         -1.
241
                         -1,
                               -1,
                                    -1,
                                          -1,
                                                -1,
                                                                       -1,
                                                                                              -1,
                                                                                                    -1,
                                                                                                          // < - -3 = non-RGB
                                                                                   -3,
       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.
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
                      CLEAR_KEYBIT(kbinput, hid_codes[i]);
             // 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++) {
                      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;
              } else if(urbinput[0] == 2)
```

```
// 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);
                                                     // stop
304
305
             CLEAR_KEYBIT(kbinput, 99);
                                                     // prev
306
             CLEAR_KEYBIT(kbinput, 100);
                                                      // play
                                                     // next
307
             CLEAR_KEYBIT(kbinput, 101);
                                                     // volup
             CLEAR_KEYBIT (kbinput, 130);
308
            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
                    SET_KEYBIT(kbinput, 99);
316
                                                     // 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
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)
             return;
for(int bit = 0; bit < 8; bit++){
   if((urbinput[0] >> bit) & 1)
339
340
341
342
                     SET_KEYBIT(kbinput, hid_codes[bit + 224]);
343
344
                     CLEAR_KEYBIT(kbinput, hid_codes[bit + 224]);
345
             for (int byte = 0; byte < 14; byte++) {
346
                 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:



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

Definition at line 366 of file keymap.c.

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

Referenced by os_inputmain().

```
367
        if ((endpoint != 2 && endpoint != -2) || length < 10)</pre>
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))</pre>
374
375
                  SET_KEYBIT(kbinput, MOUSE_BUTTON_FIRST + bit);
376
377
                  CLEAR_KEYBIT(kbinput, MOUSE_BUTTON_FIRST + bit);
378
379
         // Bytes 5 - 8: movement
         *xaxis += *(short*)(urbinput + 5);
*yaxis += *(short*)(urbinput + 7);
380
381
         // Byte 9: wheel
char wheel = urbinput[9];
382
383
384
         if(wheel > 0)
                                                                     // wheelup
385
             SET_KEYBIT(kbinput, MOUSE_EXTRA_FIRST);
386
         CLEAR_KEYBIT(kbinput, MOUSE_EXTRA_FIRST);
if(wheel < 0)</pre>
387
388
389
            SET_KEYBIT(kbinput, MOUSE_EXTRA_FIRST + 1);
                                                                     // wheeldn
         else
391
             CLEAR_KEYBIT(kbinput, MOUSE_EXTRA_FIRST + 1);
392 1
```

Here is the caller graph for this function:



5.22.4 Variable Documentation

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

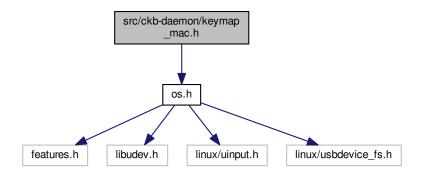
Definition at line 5 of file keymap.c.

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

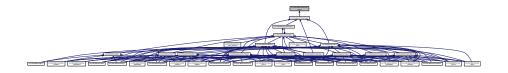
5.23 src/ckb-daemon/keymap_mac.h File Reference

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

Include dependency graph for keymap_mac.h:



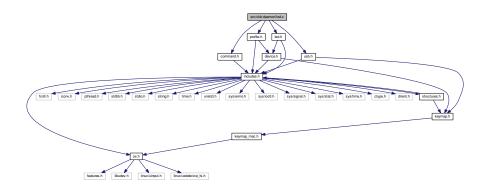
This graph shows which files directly or indirectly include this file:



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

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

Include dependency graph for led.c:



Functions

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

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

5.24.1 Function Documentation

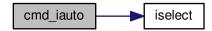
5.24.1.1 void cmd_iauto (usbdevice * kb, usbmode * mode, int dummy1, int dummy2, const char * led)

Definition at line 54 of file led.c.

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

```
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:



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

Definition at line 62 of file led.c.

References usbmode::inotify, and iselect().

```
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:



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

Definition at line 38 of file led.c.

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

```
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:



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

Definition at line 46 of file led.c.

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

```
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:



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

Definition at line 6 of file led.c.

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

```
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 }
```

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

Definition at line 73 of file led.c.

References IS_K65, IS_K95, IS_MOUSE, IS_SABRE, IS_SCIMITAR, usbdevice::product, and usbdevice::vendor. Referenced by printrgb().

```
{
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
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:



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

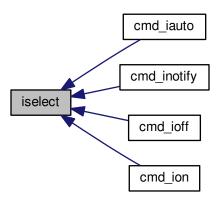
Definition at line 25 of file led.c.

References I_CAPS, I_NUM, and I_SCROLL.

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

```
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:



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

Definition at line 102 of file led.c.

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

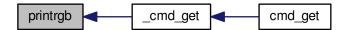
```
102
          uchar r[N_KEYS_EXTENDED], q[N_KEYS_EXTENDED], b[
103
        N_KEYS_EXTENDED];
104
          const uchar* mr = light->r;
105
          const uchar* mg = light->g;
          const uchar* mg = light->b;
const uchar* mb = light->b;
for(int i = 0; i < N_KEYS_EXTENDED; i++){
    // Translate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

```
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:



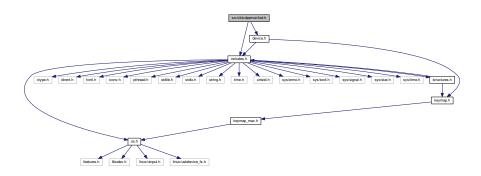
Here is the caller graph for this function:



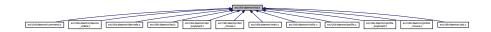
5.25 src/ckb-daemon/led.h File Reference

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

Include dependency graph for led.h:



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



Functions

- int updatergb_kb (usbdevice *kb, int force)
- int updatergb_mouse (usbdevice *kb, int force)
- int savergb_kb (usbdevice *kb, lighting *light, int mode)
- int savergb_mouse (usbdevice *kb, lighting *light, int mode)
- int loadrgb kb (usbdevice *kb, lighting *light, int mode)
- int loadrgb mouse (usbdevice *kb, lighting *light, int mode)
- char * printrgb (const lighting *light, const usbdevice *kb)
- void cmd_rgb (usbdevice *kb, usbmode *mode, int dummy, int keyindex, const char *code)
- void cmd_ioff (usbdevice *kb, usbmode *mode, int dummy1, int dummy2, const char *led)
- void cmd ion (usbdevice *kb, usbmode *mode, int dummy1, int dummy2, const char *led)
- void cmd_iauto (usbdevice *kb, usbmode *mode, int dummy1, int dummy2, const char *led)
- void cmd_inotify (usbdevice *kb, usbmode *mode, int nnumber, int dummy, const char *led)

5.25.1 Function Documentation

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

Definition at line 54 of file led.c.

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

```
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:



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

Definition at line 62 of file led.c.

References usbmode::inotify, and iselect().

```
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:



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

Definition at line 38 of file led.c.

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

```
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:



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

Definition at line 46 of file led.c.

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

```
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:



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

Definition at line 6 of file led.c.

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

```
6
7
       int index = keymap[keyindex].led;
8
       if(index < 0) {</pre>
            if (index == -2) {
                                        // Process strafe sidelights
                  uchar sideshine;
if (sscanf(code, "%2hhx",&sideshine)) // monochromatic
                        mode->light.sidelight = sideshine;
12
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
        }
```

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

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

Definition at line 181 of file led keyboard.c.

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

Referenced by hwloadmode().

```
181
        if(kb->fwversion >= 0x0120){
182
            uchar data_pkt[12][MSG_SIZE] = {
183
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
                          eturn -1;
219
220
                    // Make sure the first four bytes match
221
                    // see comment above
222
                     // if(memcmp(p, data_pkt[i + clr * 4], 4)){
                    if (memcmp(in_pkt[i], (kb->fwversion >= 0x0205)? cmp_pkt[i] : data_pkt[i + clr * 4
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
                                 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] = {
                { 0xff, 0x01, 60, 0 },
```

```
252
                     { 0xff, 0x02, 60, 0 },
                     { 0xff, 0x03, 60, 0 },
{ 0xff, 0x04, 36, 0 },
253
254
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
                          ckb_err("Bad input header\n");
264
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:



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

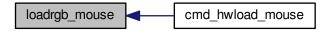
Definition at line 81 of file led mouse.c.

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

Referenced by cmd_hwload_mouse().

```
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
98
             light->g[led] = in_pkt[5];
             light->b[led] = in_pkt[6];
99
100
              // Set packet for next zone
101
              data_pkt[2]++;
102
         return 0;
103
104 }
```

Here is the caller graph for this function:



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

Definition at line 102 of file led.c.

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

```
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++) {</pre>
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
```

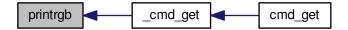
```
135
        // If they are, just output that color
136
        if(same) {
             char* buffer = malloc(7);
137
             snprintf(buffer, 7, "%02x%02x%02x", r[0], g[0], b[0]);
138
139
             return buffer;
140
        const int BUFFER_LEN = 4096;
                                           // 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:



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

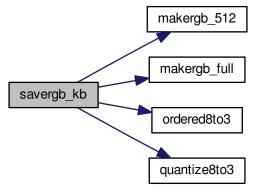
Definition at line 139 of file led keyboard.c.

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

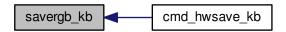
Referenced by cmd_hwsave_kb().

```
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 },
143
144
145
146
                   { 0x07, 0x14, 0x03, 0x01, 0x01, mode + 1, 0x01 },
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
170
                   { 0x7f, 0x03, 60, 0 },
171
                   { 0x7f, 0x04, 36, 0 },
172
                   { 0x07, 0x14, 0x02, 0x00, 0x01, mode + 1 }
173
174
              makergb_512(light, data_pkt, kb->dither ? ordered8to3 :
       quantize8to3);
175
              if(!usbsend(kb, data_pkt[0], 5))
176
                   return -1;
177
178
         return 0;
179 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



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

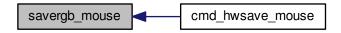
Definition at line 62 of file led_mouse.c.

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

Referenced by cmd_hwsave_mouse().

```
uchar data_pkt[MSG_SIZE] = { 0x07, 0x13, 0x10, 1, 0 };
         // Save each RGB zone, minus the DPI light which is sent in the DPI packets int zonecount = IS_SCIMITAR(kb) ? 4 : IS_SABRE(kb) ? 3 : 2;
64
65
         for(int i = 0; i < zonecount; i++) {
  int led = LED_MOUSE + i;
  if(led >= LED_DPI)
66
                    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:



5.25.1.11 int updatergb_kb (usbdevice * kb, int force)

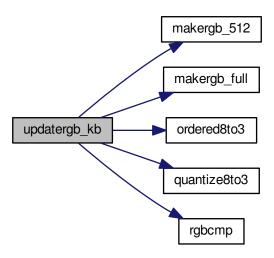
Definition at line 77 of file led_keyboard.c.

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

77

```
78
       if(!kb->active)
79
            return 0;
80
       lighting* lastlight = &kb->profile->lastlight;
       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;
86
       lastlight->forceupdate = newlight->forceupdate = 0;
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 },
116
                  \{ 0x07, 0x28, 0x03, 0x03, 0x02, 0 \}
117
             };
             makergb_full(newlight, data_pkt);
118
119
             if(!usbsend(kb, data_pkt[0], 12))
120
                 return -1;
121
        } else {
122
            // On older keyboards it looks flickery and causes lighting glitches, so we don't use it.
123
             uchar data_pkt[5][MSG_SIZE] = \{
                 { 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))
                 return -1;
132
133
134
135
        memcpy(lastlight, newlight, sizeof(lighting));
136
        return 0;
137 }
```

Here is the call graph for this function:



5.25.1.12 int updatergb_mouse (usbdevice * kb, int force)

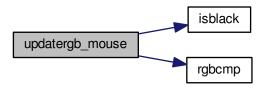
Definition at line 20 of file led_mouse.c.

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

```
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];
  *rgb_data++ = newlight->g[LED_MOUSE + i];
37
38
39
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
             data_pkt[1][4] = 1;
53
             if(!usbsend(kb, data_pkt[1], 1))
```

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

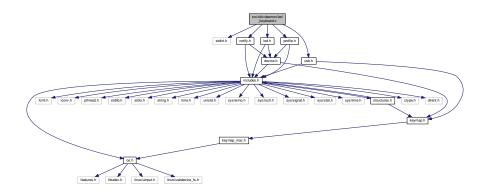
Here is the call graph for this function:



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

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

Include dependency graph for led keyboard.c:



Macros

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

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

Functions

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

Variables

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

 $0xaa) >> 1) \mid (((\ (\ (\ 0\)\ +\ 8\)\ +\ 2\)\ +\ 1\)\ \&\ 0x55) << 1))\ \&\ 0x33) << 2))\ \&\ 0x0f) << 4)), ((((((((((((((\ (\ 0\)\)\ +\ 8\)\ +\ 2\)\ +\ 1\)\)\ \&\ 0x55) << 1))))))$ (0) + (0)<< 2)) & 0xf0) >> 4) | (((((((((0) + 16) & 0xaa) >> 1) | ((((0) + 16) & 0x55) << 1)) & 0xcc) >> 2) | $1) \mid (((((((0) + 16) + 2) + 1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((((0) + 16) + 16) + 2) + 16) +$ 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)\ \big|\ \big(((\ (\ (\ 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| + 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)
((((0) + 16) + 8) + 4) + 1) & 0xaa) >> 1) | ((((((0) + 16) + 8) + 4) + 1) & 0x55) << 1)) & 0xcc)
(((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
+4)+2) & 0xaa >> 1) | ((( ( ( ( 0 ) + 16 ) + 8 ) + 4 ) + 2 ) & 0x55 << 1)) & 0xcc >> 2) | (((( ( ( ( 0 ) + 16 ) + 8 ) + 4 ) + 2 ) & 0x55
) + 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) |
<<1)) \ \& \ 0xcc)>>2) \ | \ ((((((\ (\ 0\ )\ +\ 32\ )\ \&\ 0xaa)>>1)\ |\ (((\ (\ 0\ )\ +\ 32\ )\ \&\ 0x55)<<1))\ \&\ 0x33)<<2))\ \&\ 0x33)<<2))\ \&\ 0x33)<<2)) \ \&\ 0x33)<<2)
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) $+32)+8)+4) \& 0xaa) >> 1) \mid (((((((0)+32)+8)+4) \& 0x55) << 1)) \& 0xcc) >> 2) \mid ((((((((((0)+32)+32)+32)+32)+32)+32)+32)+32)+32) + 3$ 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)0xaa >> 1 | ((((((((0) + 32) + 8) + 4) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2 | ((((((((((((0) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) (((((0)+32)+8)+4)+2)+1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 4)), (((((((((((((0)+32)+2)+2)+2)+2)+2)+2)+2)+2)+2)+2)+2)+2)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)((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))
(0.0000) > (0.0000) > (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) | (0.0000) 
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)
1) |(((((((((0)+32)+16)+4)+2)+1) & 0x55) << 1)) & 0xcc) >> 2) |((((((((((0)+32)+16)+16)+16)+16)+16)+16)+16)+16) |
((((0) + 32) + 16) + 4) + 2) + 1) & 0x55 << 1) & 0x33 << 2) & 0x0f << 4), (((((((((((((((0) + 32)) + 2) + 1) + 2) + 1) + 2) + 1) + 2) + 3))
+\ 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)
+ 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)
<<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 ) + 4 ) + 4 ) + 4 ) + 4 ) |
)+64)+4)+1) & 0xaa) >> 1) | ((((((0)+64)+4)+1) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) <<
1)) & 0xcc) >> 2 | ((((((((0)+64)+4)+4)+2) & 0xaa) >> 1 | ((((((0)+64)+4)+2) & 0x55) << 1
+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) & 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)
\$ \text{ 0xaa} >> 1 | ((((((((0)+64)+8)+4)+2) \$ \text{ 0x55}) << 1)) \$ \text{ 0xcc} >> 2 | ((((((((((0)+64)+8)+8)+8)+8)+8)+8) |
+4)+2) & Oxaa) >> 1) | ((( ( ( ( ( 0 ) +64 ) +8 ) +4 ) +2 ) & Ox55) << 1)) & Ox33) << 2)) & Ox0f) <<
0\ ) + 64\ ) + 8\ ) + 4\ ) + 2\ ) + 1\ )\ \&\ 0xaa) >> 1)\ \big|\ (((\ (\ (\ (\ 0\ ) + 64\ ) + 8\ ) + 4\ ) + 2\ ) + 1\ )\ \&\ 0x55) << 1))\ \&\ 0x55) << 1))\ \&\ 0x55) << 1)
1)) & 0xcc) >> 2) | ((((((((0)+64)+16)&0xaa)>> 1) | (((((0)+64)+16)&0x55)<< 1)) & 0x33)
0xco) >> 2) \mid ((((((((0)+64)+16) \& 0xaa) >> 1) \mid ((((((0)+64)+16) \& 0x55) << 1)) \& 0x33) << 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 > 
2) | (((((((((0)+64)+16)+2)+1)&0xaa) >> 1) | ((((((0)+64)+16)+2)+1)&0x55) << 1))
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\ )\ +\ 64\ )\ +\ 16\ )\ +\ 8\ )\ +\ 1\ )\ \&\ 0xaa)>>1)\ \big|\ (((\ (\ (\ (\ 0\ )\ +\ 64\ )\ +\ 16\ )\ +\ 8\ )\ +\ 1\ )\ \&\ 0x55)<<1))\ \&\ 0xcc)>>1)
((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)
(((0) + 64) + 16) + 8) + 4) & 0x55) << 1)) & 0xcc) >> 2) | (((((((0) + 64) + 16) + 8) + 4) & 0xaa)) | (((0) + 64) + 16) + 8) + 4) & 0xaa) | (((0) + 64) + 16) + 8) + 4) & 0xaa) | (((0) + 64) + 16) + 8) + 4) & 0xaa) | (((0) + 64) + 16) + 8) + 4) & 0xaa) | (((0) + 64) + 16) + 8) + 4) & 0xaa) | (((0) + 64) + 16) + 8) + 4) & 0xaa) | (((0) + 64) + 16) + 8) + 4) & 0xaa) | (((0) + 64) + 16) + 8) + 4) & 0xaa) | (((0) + 64) + 16) + 8) + 4) & 0xaa) | ((((0) + 64) + 16) + 8) + 4) & 0xaa) | (((0) + 64) + 16) + 8) + 4) & 0xaa) | (((0) + 64) + 16) + 8) + 4) & 0xaa) | (((0) + 64) + 16) + 8) + 4) & 0xaa) | (((0) + 64) + 16) + 8) + 4) & 0xaa) | (((0) + 64) + 16) + 8) + 4) & 0xaa) | (((0) + 64) + 16) + 8) + 4) & 0xaa) | (((0) + 64) + 16) + 8) + 4) & 0xaa) | (((0) + 64) + 16) + 8) + 4) & 0xaa) | (((0) + 64) + 16) + 8) + 4) & 0xaa) | (((0) + 64) + 16) + 8) + 4) & 0xaa) | (((0) + 64) + 16) + 8) + 4) & 0xaa) | (((0) + 64) + 16) + 8) + 4) & 0xaa) | (((0) + 64) + 16) + 8) + 4) & 0xaa) | (((0) + 64) + 16) + 8) + 4) & 0xaa) | (((0) + 64) + 16) + 8) + 4) & 0xaa) | (((0) + 64) + 16) + 8) + 4) & 0xaa) | (((0) + 64) + 16) + 8) + 4) & 0xaa) | (((0) + 64) + 16) + 8) + 4) & 0xaa) | (((0) + 64) + 16) + 8) + 4) & 0xaa) | (((0) + 64) + 16) + 8) + 4) & 0xaa) | (((0) + 64) + 16) + 8) + 4) & 0xaa) | (((0) + 64) + 16) + 8) + 4) & 0xaa) | (((0) + 64) + 16) + 8) + 4) & 0xaa) | (((0) + 64) + 16) + 8) + 4) & 0xaa) | (((0) + 64) + 16) + 8) + 4) & 0xaa) | (((0) + 64) + 16) + 8) + 4) & 0xaa) | (((0) + 64) + 16) + 8) + 4) & 0xaa) | (((0) + 64) + 16) + 8) + 4) & 0xaa) | (((0) + 64) + 16) + 8) + 4) & 0xaa) | (((0) + 64) + 16) + 8) + 4) & 0xaa) | (((0) + 64) + 16) + 8) + 4) & 0xaa) | (((0) + 64) + 16) + 8) + 4) & 0xaa) | (((0) + 64) + 16) + 8) + 4) & 0xaa) | (((0) + 64) + 16) + 8) + 4) & 0xaa) | (((0) + 64) + 16) + 8) + 4) & 0xaa) | (((0) + 64) + 16) + 8) + 4) & 0xaa) | ((0) + 64) + 8) + 4) & 0xaa) | (((0) + 64) + 8) + 4) & 0xaa) | (((0) + 64) + 8) + 4) & 0xaa) | (((0) + 64) + 8) + 4) & 0xaa) | (((0) + 64) + 8) + 4) | (((0) + 64) + 8) + 8) + 4) | (((0) + 64) +
& 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) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.016) > (0.0
(0.055) << 1) (0.055) << 1) (0.055) << 1) (0.055) << 1) (0.055) << 1) (0.055) << 1) (0.055) << 1) (0.055) << 1) (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) (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) (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) (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) (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) (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) (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) (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) << 1)
((((0) + 64) + 32) + 4) + 2) + 1) & 0x55 << 1) & 0x33 << 2) & 0x0f << 4), (((((((((((((((0) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 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) <<
1)) & 0xcc >> 2) | (((((((((0) + 64) + 32) + 8) + 2) + 1) & 0xaa) >> 1) | ((((((((0) + 64) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) 
) & 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 )
) + 64 ) + 32 ) + 16 ) & 0xaa) >> 1) | ((( ( ( ( 0 ) + 64 ) + 32 ) + 16 ) & 0x55) << 1)) & 0xcc) >> 2) | ((((( (
((0) + 64) + 32) + 16 (0) & 0xaa >> 1 ((((((0) + 64) + 32) + 16) & 0x55) << 1)) & 0x33) << 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)
 \$ \ 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)) \ \$ \ 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 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 32 ) + 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)
(0) + 64) + 32 + 16 + 8 + 1 \times 0 \times 55 < 1) \times 0 \times 55 < 2) \mid (((((((((0) + 64) + 32) + 16) + 8) + 10) \times 10) \times 10) = 1000 \times 100000 \times 10000 \times 10000 \times 1000 \times 10000 \times 1000 \times 1000 \times 1000 \times 1000 \times 1
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)\ \big|\ (((\ (\ (\ (\ (\ 0\ )\ +\ 64\ )\ +\ 32\ )\ +\ 16\ )\ +\ 8\ )\ +\ 2\ )\ +\ 1\ )\ \&\ 0x55)<<1))\ \&\ 0x33)<<2))\ \&\ 0x33)<<2))\ \&\ 0x33)<<2)
32) + 16) + 8) + 2) + 1) & 0x55 < (1)) & 0xcc) >> 2) | ((((((((0) + 64) + 32) + 16) + 8) + 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) + 16) + 16) + 16)
```

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

 $(0) + 127 + 8 \times 0$ $(0) + 127 + 8 \times 0$ $127 + 8 + 1 \times 0$ (((((((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) & 0xaa) >> 1) | (((((0) + 127) + 16) + 4) & 0x55) << 1)) & 0x33) << 2)) & 0x33 | 0x33 |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) + 127) + 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) + 127) + 16) + 8) + 127) + 16) + 127) + 16) + 127) + 16) + 127) + 16) + 127) + 16) + 127) + 16) + 127) + 16) + 127) + 16) + 127) + 16) + 127) + 16) + 127) + 16) + 127) + 16) + 127) + 16) + 127) + 16) + 127) + 16) + 127) + 16) + 127) + 16) + 127) + 16) + 127) + 16) +4) + 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) | ((0)0xaa) >> 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)+ 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\)+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|$ + 16) & 0xaa) >> 1) | ((((((0) + 127) + 32) + 16) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((0) + 127) + 32) + 16) & 0x55) << 1)) & 0xcc) >> 2) $) + 32 \) + 16 \) \ \& \ Oxaa) >> 1) \ \big| \ (((\ (\ (\ 0\) + 127\) + 32\) + 16\) \ \& \ Ox55) << 1)) \ \& \ Ox33) << 2)) \ \& \ Ox0f) << 1)$ 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) | (((((127 + 32 + 16 + 2 + 1 & 0xaa >> 1 ((((((((0) + 127) + 32) + 16) + 2) + 1) & 0x55 >< 1)) & 0xcc) >> 2 | (((((((((0) + 127) + 32) + 16) + 2) + 1) & 0xaa) >> 1) | ((((((((0) + 127) + 32) + 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) $(\;(\;0\;)\;+\;127\;)\;+\;32\;)\;+\;16\;)\;+\;4\;)\;+\;2\;)\;\&\;0xaa)>>\;1)\;|\;(((\;(\;(\;0\;)\;+\;127\;)\;+\;32\;)\;+\;16\;)\;+\;4\;)\;+\;2\;)\;\&\;0x55)$

16) + 4) + 2) + 1) & 0xaa) >> 1) | (((((((((0) + 127) + 32) + 16) + 4) + 2) + 1) & 0x55) << 1)) & 0x55) <= 1) & 0x55) << 1)) & 0x55) <= 1) & 0x55) <= 1)+ 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) <<+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) << $+\ 4\)\ +\ 2\)\ +\ 1\)\ \&\ 0xaa) >>\ 1)\ \big|\ (((\ (\ (\ (\ (\ (\ 0\)\ +\ 127\)\ +\ 32\)\ +\ 16\)\ +\ 8\)\ +\ 4\)\ +\ 2\)\ +\ 1\)\ \&\ 0x55) <<\ 1))\ \&\ 0x55) <<\ 1))\ \&\ 0x55$ $\$ \text{ 0xaa} >>> 1) \mid (((\ (\ 0\) + 127\) + 64\) \ \$ \text{ 0x55}) << 1)) \ \$ \text{ 0x33}) << 2)) \ \$ \text{ 0x0f}) << 4)), ((((((((\ (\ 0\) + 127\) + 64\) \ \$ \text{ 0x55}) << 1)) \ \$ \text{ 0x33}) << 2)) \ \$ \text{ 0x0f}) << 4)), (((((((\ (\ 0\) + 127\) + 64\) \ \$ \text{ 0x55}) << 1)) \ \$ \text{ 0x33}) << 2)) \ \$ \text{ 0x0f}) << 4)), (((((((\ (\ 0\) + 127\) + 64\) \ \$ \text{ 0x55}) << 1)) \ \$ \text{ 0x33}) << 2)) \ \$ \text{ 0x0f}) << 4)), ((((((\ (\ 0\) + 127\) + 64\) \ \$ \text{ 0x55}) << 1)) \ \$ \text{ 0x33}) << 2)) \ \$ \text{ 0x0f}) << 4)), (((((\ (\ 0\) + 127\) + 64\) \ \$ \text{ 0x55}) << 1)) \ \$ \text{ 0x33}) << 2)) \ \$ \text{ 0x0f}) << 4)), ((((\ (\ (\ 0\) + 127\) + 64\) \ \$ \text{ 0x55}) << 1)) \ \$ \text{ 0x33}) << 2)) \ \$ \text{ 0x0f}) << 4)), (((\ (\ (\ 0\) + 127\) + 64\) \ \$ \text{ 0x55}) << 1)) \ \$ \text{ 0x33}) << 2)) \ \$ \text{ 0x0f}) << 4)), ((\ (\ (\ (\ 0\) + 127\) + 64\) \ \$ \text{ 0x55}) << 1)) \ \$ \text{ 0x55}) << 1)$ + 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) >> 0)$ $(\ (\ (\ 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\)\ +\ 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)

```
) + 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) >>
(\;(\;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) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16
127 + 64 + 16 + 8 + 4 & 0xaa >> 1 | ((( ( ( ( ( ( 0 ) + 127 ) + 64 ) + 16 ) + 8 ) + 4 ) & 0x55 >< 1) \ & 0x55 >< 1
(0.0033) << 2) & 0.0000 >> 4 
+ 127 ) + 64 ) + 16 ) + 8 ) + 4 ) & 0x55) << 1)) & 0xcc) >> 2) | ((((( ( ( ( ( ( 0 ) + 127 ) + 64 ) + 16 ) + 8 ) + 4 ) + 16 ) + 8 ) + 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) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 16) + 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
+2)+1) & 0x55) <<1)) & 0x33) <<2)) & 0x0f) <<4)), ((((((((((0)+127)+64)+32) & 0xaa) >> 127) + 64) + 32) & 0xaa) >> 127) + 64) + 32) & 0xaa) >> 127) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) + 64) 
127) + 64) + 32) & 0xaa) >> 1) | ((((((0) + 127) + 64) + 32) & 0x55) << 1)) & 0x33) << 2)) & 0x0f)
1) \& 0xaa) >> 1) \mid (((((((0) + 127) + 64) + 32) + 1) \& 0x55) << 1)) \& 0xcc) >> 2) \mid (((((((((((0) + 127) + 127) + 64) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 127) + 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) + 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) << 100
((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) (0.0x55) << 1) (0.0x55) << 1)
8) & 0xaa) >> 1) | (((((((0) + 127) + 64) + 32) + 8) & <math>0x55) << 1)) & <math>0x33) << 2)) & <math>0x0f) << 4)),
0xaa) >> 1 | ((( ( ( ( ( ( ( 0 ) + 127 ) + 64 ) + 32 ) + 8 ) + 2 ) & 0x55) << 1)) & 0xcc) >> 2 | ((((( ( ( ( ( ( 0 ) + 127 ) + 64 ) + 32 ) + 8 ) + 2 ) & 0x55) << 1))
+ 127) + 64) + 32) + 8) + 2) & 0x55) << 1)) & 0xcc) >> 2) | (((((((((0) + 127) + 64) + 32) + 8) + 2)
32) + 8) + 2) + 1) & 0x55) << 1)) & 0xcc) >> 2) | ((((((((((0) + 127) + 64) + 32) + 8) + 2) + 1)
(0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.001) < (0.0
```

```
+4)+1) & 0xaa) >> 1) | ((((((((0)+127)+64)+32)+8)+4)+1) & 0x55) << 1)) & 0xcc) >> 2)
>> 1) | ((( ( ( ( ( ( ( ( ( ( 0 ) + 127 ) + 64 ) + 32 ) + 8 ) + 4 ) + 2 ) + 1 ) & 0x55) << 1)) & 0x33) << 2)) & 0x0f)
16\ )\ \&\ 0xaa) >> 1)\ |\ (((\ (\ (\ 0\ )\ +\ 127\ )\ +\ 64\ )\ +\ 32\ )\ +\ 16\ )\ \&\ 0x55) << 1))\ \&\ 0xcc) >> 2)\ |\ (((((\ (\ (\ 0\ )\ )\ +\ 127\ )\ +\ 64\ )\ +\ 32\ )\ +\ 16\ )\ \&\ 0x55) << 1))\ \&\ 0xcc) >> 2)\ |\ (((((\ (\ (\ 0\ )\ )\ +\ 127\ )\ +\ 64\ )\ +\ 32\ )\ +\ 16\ )\ \&\ 0x55) << 1))\ \&\ 0xcc) >> 2)\ |\ (((((\ (\ (\ 0\ )\ )\ +\ 127\ )\ +\ 16\ )\ +\ 16\ )\ )\ \&\ 0x55) << 1))\ \&\ 0xcc) >> 2)\ |\ (((((\ (\ (\ 0\ )\ )\ +\ 127\ )\ +\ 16\ )\ )\ &\ 0x55) << 1))\ \&\ 0xcc) >> 2)\ |\ (((((\ (\ (\ 0\ )\ )\ +\ 127\ )\ +\ 16\ )\ )\ &\ 0x55) << 1))\ &\ 0xcc) >> 2)\ |\ (((((\ (\ (\ 0\ )\ )\ +\ 127\ )\ +\ 127\ )\ +\ 16\ )\ )\ &\ 0x55) << 1)\ |\ 0xcc) >> 2)\ |\ ((((\ (\ (\ (\ 0\ )\ )\ +\ 127\ )\ +\ 127\ )\ +\ 16\ )\ )\ &\ 0x55) << 1)\ |\ 0xcc) >> 2)\ |\ ((((\ (\ (\ (\ 0\ )\ )\ +\ 127\ )\ +\ 127\ )\ +\ 16\ )\ )\ |\ 0x55]
+ 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) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 32) + 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))\ \& (x + 10)\ (
(((((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)
) + 127 + 64 + 32 + 16 + 4 + 2 & 0xaa >> 1 | ((((((((0) + 127) + 64) + 32) + 16) + 4) + 2)
(((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) <<
127 + 64 + 32 + 16 + 8 + 2 \times 8  0xaa) >> 1) | ((((((((0) + 127) + 64) + 32) + 16) + 8) + 2) \times 8 
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)\ \big|\ (((\ (\ (\ (\ (\ (\ 0\ )\ +\ 127\ )\ +\ 64\ )\ +\ 32\ )\ +\ 16\ )\ +\ 8\ )\ +\ 4\ )\ \&\ 0x55)<<1))\ \&\ 0xcc)>>2)\ \big|
)+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) & 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) + 1
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)), }
```

5.26.1 Macro Definition Documentation

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

Definition at line 9 of file led_keyboard.c.

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

Definition at line 10 of file led keyboard.c.

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

Definition at line 11 of file led_keyboard.c.

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

Definition at line 12 of file led_keyboard.c.

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

Definition at line 13 of file led_keyboard.c.

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

Definition at line 14 of file led_keyboard.c.

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

Definition at line 15 of file led_keyboard.c.

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

Definition at line 16 of file led_keyboard.c.

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

Definition at line 17 of file led_keyboard.c.

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

Definition at line 18 of file led_keyboard.c.

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

Definition at line 19 of file led_keyboard.c.

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

Definition at line 20 of file led_keyboard.c.

5.26.2 Function Documentation

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

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

Definition at line 181 of file led_keyboard.c.

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

Referenced by hwloadmode().

```
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;
                     // Make sure the first four bytes match
220
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 *
       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]);
                         in_pkt[2][0] = 0x99;

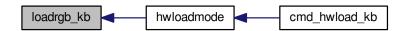
in_pkt[2][1] = 0x99;
229
230
231
                         in_pkt[2][2] = 0x99;
232
                         in_pkt[2][3] = 0x99;
233
                         usbrecv(kb, in_pkt[2], in_pkt[2]); // just to find it in the wireshark log
234
                         return -1;
235
236
237
                 // 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] +
271
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:



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

Definition at line 36 of file led_keyboard.c.

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

Referenced by savergb_kb(), and updatergb_kb().

```
39
          // Compress RGB values to a 512-color palette
40
          for (int i = 0; i < N_KEYS_HW; i += 2) {</pre>
41
                char g1 = ditherfn(i, light->g[i]), g2 = ditherfn(i + 1, light->g[i + 1]); char b1 = ditherfn(i, light->b[i]), b2 = ditherfn(i + 1, light->b[i + 1]); r[i / 2] = (7 - r2) << 4 | (7 - r1); g[i / 2] = (7 - g2) << 4 | (7 - g1);
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:



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

Definition at line 56 of file led_keyboard.c.

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

Referenced by savergb_kb(), and updatergb_kb().

```
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:



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

Definition at line 24 of file led_keyboard.c.

References bit_reverse_table.

Referenced by savergb_kb(), and updatergb_kb().

Here is the caller graph for this function:



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

Definition at line 32 of file led_keyboard.c.

Referenced by savergb_kb(), and updatergb_kb().

Here is the caller graph for this function:



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

Definition at line 72 of file led_keyboard.c.

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

Referenced by updatergb_kb().

Here is the caller graph for this function:



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

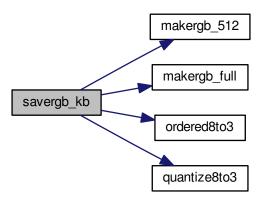
Definition at line 139 of file led_keyboard.c.

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

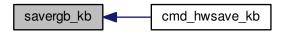
Referenced by cmd_hwsave_kb().

```
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, mode + 1, 0x01 },
145
146
147
                 // Green
                 { 0x7f, 0x01, 60, 0 },
{ 0x7f, 0x02, 60, 0 },
148
149
150
                 { 0x7f, 0x03, 24, 0 },
151
                 { 0x07, 0x14, 0x03, 0x01, 0x01, mode + 1, 0x02 },
152
                 // Blue
                 { 0x7f, 0x01, 60, 0 },
153
154
                 { 0x7f, 0x02, 60, 0 },
155
                 { 0x7f, 0x03, 24, 0 },
                 { 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:



5.26.2.8 int updatergb_kb (usbdevice * kb, int force)

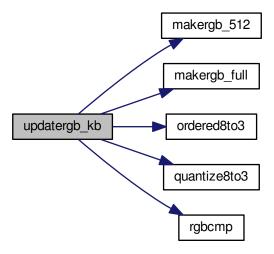
Definition at line 77 of file led_keyboard.c.

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

```
78
      if(!kb->active)
79
           return 0;
      lighting* lastlight = &kb->profile->lastlight;
lighting* newlight = &kb->profile->currentmode->
80
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
97
                   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);
119
              if(!usbsend(kb, data_pkt[0], 12))
                   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:



5.26.3 Variable Documentation

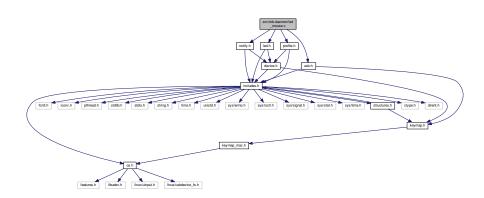
1) | (((0) & 0x55) << 1)) & 0x33) << 2)) & 0xf0) >> 4) | (((((((0) & 0x55) << 1)) & 0xcc) >> $0\)+1\)\ \&\ 0x55)<<1))\ \&\ 0xcc)>>2)\ |\ ((((((\ 0\)+1\)\ \&\ 0xaa)>>1)\ |\ ((((\ 0\)+1\)\ \&\ 0x55)<<1))\ \&\ 0x33)<<2))\ \&\ 0x33)<<2))\ \&\ 0x33)<<2))\ \&\ 0x33)<<2)$ 1) | ((((0)+1)&0x55) << 1)) & 0x33) << 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\)\)\ $) \& 0x55) <<1)) \& 0xcc)>>2) \mid (((((((0)+4)+1) \& 0xaa)>>1) \mid (((((0)+4)+1) \& 0x55) <<1)) \& 0x33) <<1)$ +4) +1) & 0xaa) >>1) | (((((0)+4)+1) & 0x55) <<1)) & 0x33) <<2)) & 0x0f) <<4), ((((((((((((((0)+4)+2)) & 0x55) & 0x0f) & 0x0f) & 0x0f) & 0x0f) & 0x0f) 1)) & 0xcc) >> 2) | (((((((0)+4)+2) & 0xaa) >> 1) | (((((0)+4)+2) & 0x5) << 1)) & 0x33) << 2)) & 0x0f) << $) + 2 + 1 + 3 \times 3$ 0xaa) $>> 1 + 2 \times 4 \times 5$ 0xaa) $>> 1 + 2 \times 4 \times 5$ 0xcc) $>> 2 \times 4 \times 5$ 0xaa)) + 8) & 0x55) << 1)) & 0xcc) >> 2) | (((((((0) + 8) & 0xaa) >> 1) | ((((0) + 8) & 0x55) << 1)) & 0x33) << 2)) & 0xf0 >> 4) | ((((((((0)+8)&0xaa)>> 1) | ((((0)+8)&0x55)<<1)) & 0xcc) >> 2) | ((((((0)+8)&0xaa)>> 1) | ((((0)+8)&0x55)<<1)) & 0xcc) >> 2) | (((((0)+8)&0xaa)>> 1) | ((((0)+8)&0xaa)>> 1) | ((((0)+8)&0xaa)> 1) | ((((0)+8)&0xaa)>> 1) | (((((0)+8)&0xaa)>>) & 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) & (10) & 1)) & 0xcc) >> 2) | (((((((0)+8)+2) & 0xaa) >> 1) | (((((0)+8)+2) & 0x55) << 1)) & 0x33) << 2)) & 0x0f) << 1 $>>1) \mid (((\ (\ (\ (\ 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) $2) \mid (((((((0)+16)+1) \& 0xaa) >> 1) \mid (((((0)+16)+1) \& 0x55) << 1)) \& 0x33) << 2)) \& 0xf0) >> 4) \mid (((((((0)+16)+1) \& 0x55) << 1)) \& 0x33) << 2)) \& 0xf0) >> 4) \mid ((((((((0)+16)+1) \& 0x55) << 1)) \& 0x55) << 1)) \& 0x55) << 1) & 0x55 << 1)$ $\frac{16) + 2) & 0x55}{((((((0) + 16) + 2) & 0xaa) >> 1) | (((((0) + 16) + 2) & 0x55) << 1)) & -2) & 0x55) << 1) | ((((0) + 16) + 2) & 0x55) << 1) | (((0) + 16) + 2) & 0x55) << 1) | (((0) + 16) + 2) & 0x55) << 1) | (((0) + 16) + 2) & 0x55) << 1) | (((0) + 16) + 2) & 0x55) << 1) | (((0) + 16) + 2) & 0x55) << 1) | (((0) + 16) + 2) & 0x55) << 1) | (((0) + 16) + 2) & 0x55) << 1) | (((0) + 16) + 2) & 0x55) << 1) | (((0) + 16) + 2) & 0x55) << 1) | (((0) + 16) + 2) & 0x55) << 1) | (((0) + 16) + 2) & 0x55) << 1) | ((0) + 16) + 2) & 0x55) << 1) | ((0) + 16) + 2) & 0x55) << 1) | ((0) + 16) + 2) & 0x55) << 1) | ((0) + 16) + 2) & 0x55) << 1) | ((0) + 16) + 2) & 0x55) << 1) | ((0) + 16) + 2) & 0x55) << 1) | (0) + 16) + 2) & 0x55) << 1) | (0) + 16) + 2) & 0x55) << 1) | (0) + 16) + 2) & 0x55) << 1) | (0) + 16) + 2) & 0x55) << 1) | (0) + 16) + 2) & 0x55) << 1) | (0) + 16) + 2) & 0x55$ $(0) + 16 + 2 + 1 \times 0$ $(0) + 16 + 2 + 1 \times 0$ $(0) + 16 + 2 + 1 \times 0$ $(0) + 16 + 2 \times 0$ $(0) + 16 \times 0$ (0) +

Referenced by ordered8to3().

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

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

Include dependency graph for led_mouse.c:



Functions

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

5.27.1 Function Documentation

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

Definition at line 13 of file led mouse.c.

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

Referenced by updatergb_mouse().

Here is the caller graph for this function:



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

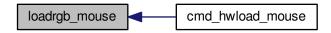
Definition at line 81 of file led_mouse.c.

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

Referenced by cmd_hwload_mouse().

```
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
                 led++;
                                    // Skip DPI light
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:



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

Definition at line 7 of file led_mouse.c.

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

Referenced by updatergb_mouse().

Here is the caller graph for this function:



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

Definition at line 62 of file led_mouse.c.

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

Referenced by cmd_hwsave_mouse().

```
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;
63
64
65
          for(int i = 0; i < zonecount; i++) {</pre>
              int led = LED_MOUSE + i;
if(led >= LED_DPI)
68
                      led++;
                                              // Skip DPI light
69
70
               data_pkt[4] = light->r[led];
               data_pkt[1] = light->g[led];
data_pkt[6] = light->b[led];
               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:



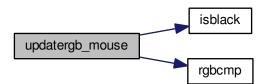
5.27.1.5 int updatergb_mouse (usbdevice * kb, int force)

Definition at line 20 of file led_mouse.c.

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

```
20
2.1
       if(!kb->active)
2.2
           return 0;
       lighting* lastlight = &kb->profile->lastlight;
23
       lighting* newlight = &kb->profile->currentmode->
24
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
       memcpy(lastlight, newlight, sizeof(lighting));
58
59
       return 0:
60 }
```

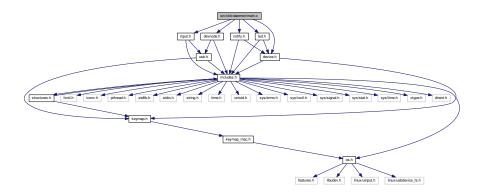
Here is the call graph for this function:



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

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

Include dependency graph for main.c:



Functions

- static void quitWithLock (char mut)
 - quitWithLock
- int restart ()
- void timespec_add (struct timespec *timespec, long nanoseconds)
- static void quit ()

quit Stop working the daemon. function is called if the daemon received a sigterm In this case, locking the device-mutex is ok.

- void sighandler2 (int type)
- void sighandler (int type)
- void localecase (char *dst, size_t length, const char *src)
- int main (int argc, char **argv)

Variables

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

brief .

· int features mask

brief.

int hwload_mode

5.28.1 Function Documentation

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

Definition at line 71 of file main.c.

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

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

Definition at line 88 of file main.c.

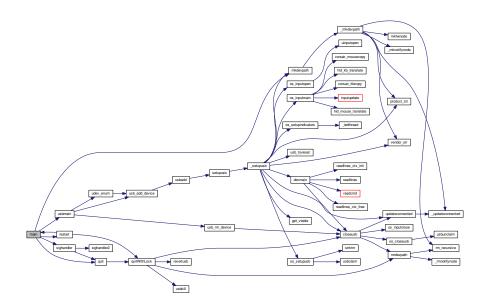
References ckb_fatal_nofile, ckb_info, ckb_info_nofile, ckb_warn_nofile, devpath, FEAT_BIND, FEAT_MOUSE-ACCEL, FEAT_NOTIFY, features_mask, gid, hwload_mode, keyboard, main_ac, main_av, mkdevpath(), quit(), restart(), sighandler(), and usbmain().

Referenced by restart().

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

```
again) \n", pidpath);
146
                     return 0;
147
                 }
148
149
150
151
        // Read parameters
152
         int forceroot = 1;
153
        for(int i = 1; i < argc; i++) {</pre>
154
             char* argument = argv[i];
             unsigned newgid;
155
156
             char hwload[7];
157
             if (sscanf(argument, "--gid=%u", &newgid) == 1) {
158
                 // Set dev node GID
159
                 gid = newgid;
            ckb_info_nofile("Setting /dev node gid: %u\n", newgid);
} else if(!strcmp(argument, "--nobind")){
   // Disable key notifications and rebinding
   features_mask &= ~FEAT_BIND & ~FEAT_NOTIFY;
160
161
162
163
                 ckb_info_nofile("Key binding and key notifications are disabled\n");
164
165
             } else if(!strcmp(argument, "--nonotify")){
166
                 // Disable key notifications
167
                 features_mask &= ~FEAT_NOTIFY;
             ckb_info_nofile("Key notifications are disabled\n");
} else if(sscanf(argument, "--hwload=%6s", hwload) == 1){
168
169
                  if(!strcmp(hwload, "always") || !strcmp(hwload, "yes") || !strcmp(hwload, "y") || !strcmp(
170
      hwload, "a")){
      171
172
173
                     hwload_mode = 1;
ckb_info_nofile("Setting hardware load: tryonce\n");
174
175
176
                 } else if(!strcmp(hwload, "never") || !strcmp(hwload, "none") || !strcmp(hwload, "no") || !
      strcmp(hwload, "n")){
177
                     hwload mode = 0:
                     ckb_info_nofile("Setting hardware load: never\n");
178
179
180
             } else if(!strcmp(argument, "--nonroot")){
181
                 // Allow running as a non-root user
182
                 forceroot = 0;
183
184 #ifdef OS MAC
185
            else if(!strcmp(argument, "--nomouseaccel")){
                // On OSX, provide an option to disable mouse acceleration
187
                 features_mask &= ~FEAT_MOUSEACCEL;
                 ckb_info_nofile("Mouse acceleration disabled\n");
188
189
             }
190 #endif
191
192
193
         // Check UID
194
        if (getuid() != 0) {
195
             if (forceroot) {
                 ckb_fatal_nofile("ckb-daemon must be run as root. Try 'sudo %s'\n", argv[0]);
196
197
                 exit(0);
198
             } else
199
                 ckb_warn_nofile("Warning: not running as root, allowing anyway per command-line
200
201
        // Make root keyboard
202
203
        umask(0);
204
        memset(keyboard, 0, sizeof(keyboard));
205
         if (!mkdevpath (keyboard))
206
             ckb_info("Root controller ready at s0\n", devpath);
207
208
        // Set signals
209
        sigset t signals:
210
        sigfillset(&signals);
211
        sigdelset(&signals, SIGTERM);
212
         sigdelset(&signals, SIGINT);
213
        sigdelset(&signals, SIGQUIT);
214
        sigdelset(&signals, SIGUSR1);
// Set up signal handlers for quitting the service.
215
216
        sigprocmask(SIG_SETMASK, &signals, 0);
217
        signal(SIGTERM, sighandler);
        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 call graph for this function:



Here is the caller graph for this function:



```
5.28.1.3 static void quit( ) [static]
```

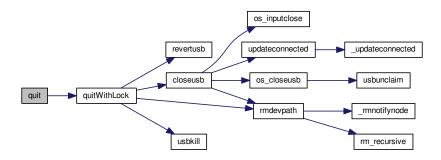
Definition at line 30 of file main.c.

References quitWithLock().

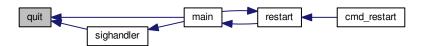
Referenced by main(), and sighandler().

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

Here is the call graph for this function:



Here is the caller graph for this function:



5.28.1.4 void quitWithLock (char mut) [static]

Parameters

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

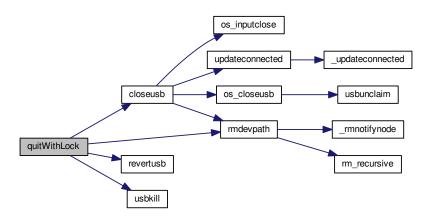
Definition at line 40 of file main.c.

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

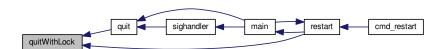
Referenced by quit(), and restart().

```
40
         // Abort any USB resets in progress
41
         freset_stop = 1;
for(int i = 1; i < DEV_MAX; i++) {
    // Before closing, set all keyboards back to HID input mode so that the stock driver can still talk</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
55 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



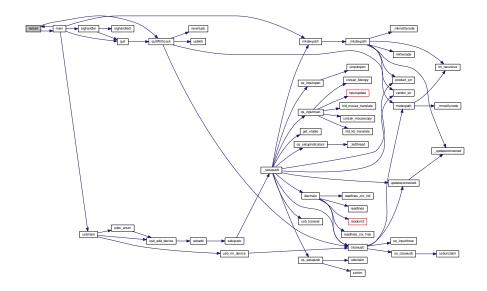
```
5.28.1.5 int restart ( )
```

Definition at line 228 of file main.c.

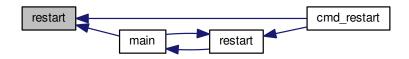
References ckb_err, main(), main_ac, main_av, and quitWithLock().

Referenced by cmd_restart(), and main().

Here is the call graph for this function:



Here is the caller graph for this function:



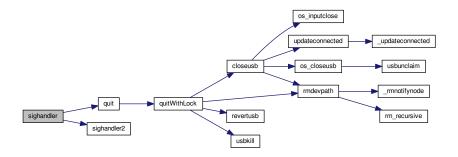
5.28.1.6 void sighandler (int type)

Definition at line 62 of file main.c.

References quit(), and sighandler2().

Referenced by main().

Here is the call graph for this function:



Here is the caller graph for this function:



5.28.1.7 void sighandler2 (int type)

Definition at line 57 of file main.c.

Referenced by sighandler().

```
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:



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

Definition at line 19 of file main.c.

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

5.28.2 Variable Documentation

```
5.28.2.1 int features mask
```

features_mask Mask of features to exclude from all devices

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

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

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

Definition at line 35 of file usb.c.

Referenced by _setupusb(), and main().

```
5.28.2.2 int hwload_mode
```

Definition at line 7 of file device.c.

Referenced by main().

```
5.28.2.3 int main_ac [static]
```

Definition at line 7 of file main.c.

Referenced by main(), and restart().

```
5.28.2.4 char** main_av [static]
```

Definition at line 8 of file main.c.

Referenced by main(), and restart().

```
5.28.2.5 volatile int reset_stop
```

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

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

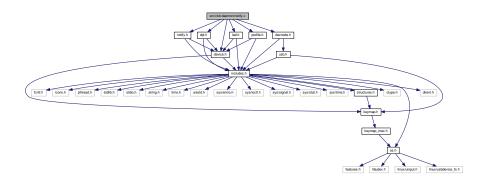
Definition at line 25 of file usb.c.

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

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

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

Include dependency graph for notify.c:



Macros

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

Functions

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

5.29.1 Macro Definition Documentation

5.29.1.1 #define HW_STANDARD

Value:

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

Definition at line 83 of file notify.c.

Referenced by _cmd_get().

5.29.1.2 #define HWMODE_OR_RETURN(kb, index)

Value:

Definition at line 73 of file notify.c.

5.29.2 Function Documentation

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

Definition at line 90 of file notify.c.

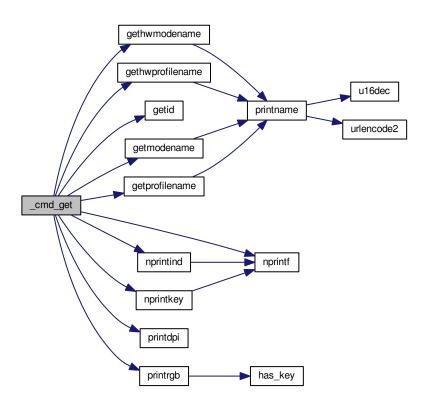
References dpiset::current, usbmode::dpi, hwprofile::dpi, gethwmodename(), gethwprofilename(), getid(), getmodename(), getprofilename(), usbdevice::hw_ileds, HW_STANDARD, I_CAPS, I_NUM, I_SCROLL, usbmode::id, usbprofile::id, hwprofile::id, usbdevice::input, keymap, usbinput::keys, dpiset::lift, usbmode::light, 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
93
           // Get the current mode number
94
           nprintf(kb, nnumber, mode, "switch\n");
95
           return:
96
      } else if(!strcmp(setting, ":rqb")){
          // Get the current RGB settings
98
           char* rgb = printrgb(&mode->light, kb);
99
           nprintf(kb, nnumber, mode, "rgb %s\n", rgb);
100
            free (rqb);
101
            return;
        } else if(!strcmp(setting, ":hwrgb")){
102
103
            // Get the current hardware RGB settings
104
            HW_STANDARD;
105
            char* rgb = printrgb(kb->hw->light + index, kb);
106
            nprintf(kb, nnumber, mode, "hwrgb %s\n", rgb);
107
            free (rqb);
108
            return;
       } else if(!strcmp(setting, ":profilename")){
109
            // 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);
       } else if(!strcmp(setting, ":hwprofilename")){
    // Get the current hardware profile name
119
120
121
            if(!kb->hw)
122
            char* name = gethwprofilename(kb->hw);
123
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
            char* name = gethwmodename(kb->hw, index);
129
130
            nprintf(kb, nnumber, mode, "hwname %s\n", name[0] ? name : "Unnamed");
131
            free(name);
132
        } else if(!strcmp(setting, ":profileid")){
           // Get the current profile ID
133
134
            char* guid = getid(&profile->id);
135
            int modified;
136
            memcpy(&modified, &profile->id.modified, sizeof(modified));
137
            nprintf(kb, nnumber, 0, "profileid %s %x\n", guid, modified);
138
            free (quid);
139
        } else if(!strcmp(setting, ":id")){
            // Get the current mode ID
140
            char* guid = getid(&mode->id);
141
142
            int modified;
143
            memcpy(&modified, &mode->id.modified, sizeof(modified));
            nprintf(kb, nnumber, mode, "id %s %x\n", guid, modified);
144
            free (quid);
145
146
        } else if(!strcmp(setting, ":hwprofileid")){
           // Get the current hardware profile ID
147
            if(!kb->hw)
149
                return;
150
            char* guid = getid(&kb->hw->id[0]);
151
            int modified:
            memcpy(&modified, &kb->hw->id[0].modified, sizeof(modified));
152
            nprintf(kb, nnumber, 0, "hwprofileid %s %x\n", guid, modified);
```

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

Here is the call graph for this function:



Here is the caller graph for this function:



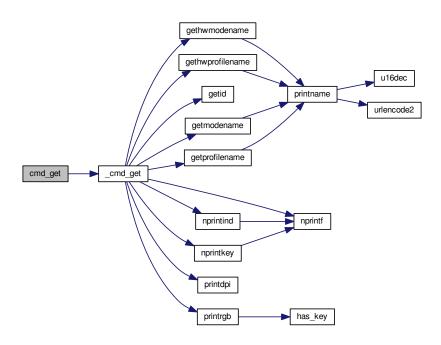
5.29.2.2 void cmd_get (usbdevice * kb, usbmode * mode, int nnumber, int dummy, const char * setting)

Definition at line 215 of file notify.c.

References _cmd_get(), and imutex.

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

Here is the call graph for this function:



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

Definition at line 61 of file notify.c.

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

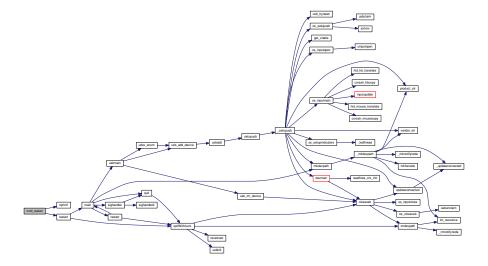
```
61
62    if(keyindex >= N_KEYS_INPUT)
63        return;
64    pthread_mutex_lock(imutex(kb));
65    if(!strcmp(toggle, "on") || *toggle == 0)
66        SET_KEYBIT(mode->notify[nnumber], keyindex);
67    else if(!strcmp(toggle, "off"))
68        CLEAR_KEYBIT(mode->notify[nnumber], keyindex);
69  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

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

Definition at line 223 of file notify.c.

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

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



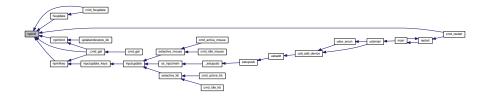
5.29.2.5 void nprintf (usbdevice * kb, int nodenumber, usbmode * mode, const char * format, ...)

Definition at line 8 of file notify.c.

References INDEX_OF, usbprofile::mode, usbdevice::outfifo, OUTFIFO_MAX, and usbdevice::profile. Referenced by _cmd_get(), cmd_fwupdate(), cmd_restart(), fwupdate(), nprintind(), and nprintkey().

```
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:



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

Definition at line 43 of file notify.c.

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

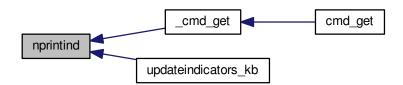
Referenced by _cmd_get(), and updateindicators_kb().

```
43
44
       const char* name = 0;
       switch(led){
45
       case I_NUM:
           name = "num";
47
48
      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:



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

Definition at line 35 of file notify.c.

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

Referenced by _cmd_get(), and inputupdate_keys().

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



Here is the caller graph for this function:



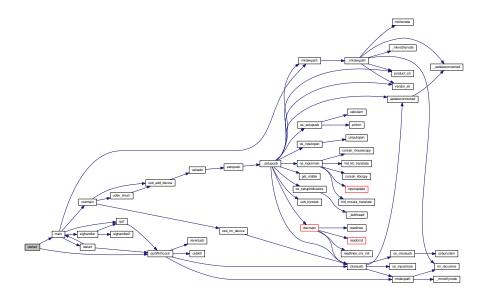
```
5.29.2.8 int restart ( )
```

Definition at line 228 of file main.c.

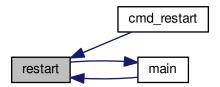
References ckb_err, main(), main_ac, main_av, and quitWithLock().

Referenced by cmd_restart(), and main().

Here is the call graph for this function:

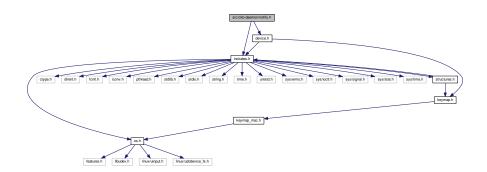


Here is the caller graph for this function:



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

#include "includes.h"
#include "device.h"
Include dependency graph for notify.h:



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



Functions

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

5.30.1 Function Documentation

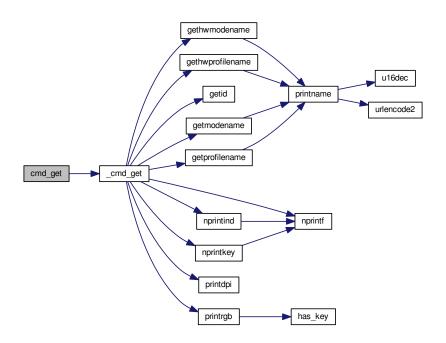
5.30.1.1 void cmd_get (usbdevice * kb, usbmode * mode, int nnumber, int dummy, const char * setting)

Definition at line 215 of file notify.c.

References _cmd_get(), and imutex.

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

Here is the call graph for this function:



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

Definition at line 61 of file notify.c.

References CLEAR KEYBIT, imutex, N KEYS INPUT, usbmode::notify, and SET KEYBIT.

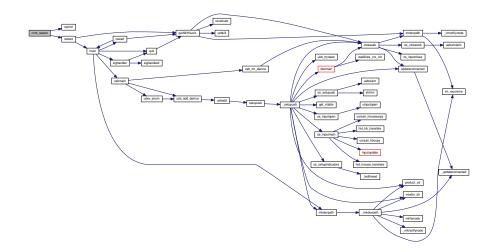
```
61
62
       if(keyindex >= N_KEYS_INPUT)
63
           return;
       pthread_mutex_lock(imutex(kb));
64
       if(!strcmp(toggle, "on") || *toggle == 0)
66
           SET_KEYBIT(mode->notify[nnumber], keyindex);
67
       else if(!strcmp(toggle, "off"))
68
          CLEAR_KEYBIT (mode->notify[nnumber],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

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

Definition at line 223 of file notify.c.

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

Here is the call graph for this function:



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

Definition at line 8 of file notify.c.

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

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

```
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;
2.3
        // 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 }
```



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

Definition at line 43 of file notify.c.

References I CAPS, I NUM, I SCROLL, and nprintf().

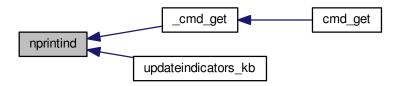
Referenced by _cmd_get(), and updateindicators_kb().

```
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
57
      nprintf(kb, nnumber, 0, "i %c%s\n", on ? '+' : '-', name);
58
59 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



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

Definition at line 35 of file notify.c.

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

Referenced by _cmd_get(), and inputupdate_keys().

Here is the call graph for this function:



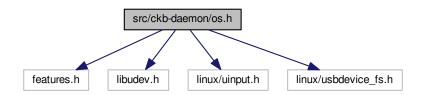
Here is the caller graph for this function:



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

#include <features.h>

```
#include <libudev.h>
#include <linux/uinput.h>
#include <linux/usbdevice_fs.h>
Include dependency graph for os.h:
```



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



Macros

- #define _DEFAULT_SOURCE
- #define GNU SOURCE
- #define UINPUT_VERSION 2
- #define euid_guard_start
- #define euid_guard_stop

5.31.1 Macro Definition Documentation

5.31.1.1 #define _DEFAULT_SOURCE

Definition at line 22 of file os.h.

5.31.1.2 #define _GNU_SOURCE

Definition at line 26 of file os.h.

5.31.1.3 #define euid_guard_start

Definition at line 40 of file os.h.

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

5.31.1.4 #define euid_guard_stop

Definition at line 41 of file os.h.

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

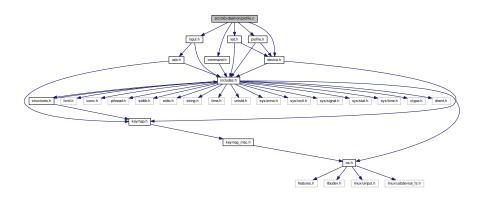
5.31.1.5 #define UINPUT_VERSION 2

Definition at line 35 of file os.h.

5.32 src/ckb-daemon/profile.c File Reference

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

Include dependency graph for profile.c:



Functions

- void urldecode2 (char *dst, const char *src)
- void urlencode2 (char *dst, const char *src)
- int setid (usbid *id, const char *guid)
- char * getid (usbid *id)
- void u16enc (char *in, ushort *out, size t *srclen, size t *dstlen)
- void u16dec (ushort *in, char *out, size_t *srclen, size_t *dstlen)
- void cmd_name (usbdevice *kb, usbmode *mode, int dummy1, int dummy2, const char *name)
- void cmd_profilename (usbdevice *kb, usbmode *dummy1, int dummy2, int dummy3, const char *name)
- char * printname (ushort *name, int length)
- char * getmodename (usbmode *mode)
- char * getprofilename (usbprofile *profile)
- char * gethwmodename (hwprofile *profile, int index)
- char * gethwprofilename (hwprofile *profile)
- void cmd_id (usbdevice *kb, usbmode *mode, int dummy1, int dummy2, const char *id)
- void cmd_profileid (usbdevice *kb, usbmode *mode, int dummy1, int dummy2, const char *id)
- static void initmode (usbmode *mode)
- void allocprofile (usbdevice *kb)
- int loadprofile (usbdevice *kb)
- static void freemode (usbmode *mode)
- void cmd_erase (usbdevice *kb, usbmode *mode, int dummy1, int dummy2, const char *dummy3)
- static void _freeprofile (usbdevice *kb)
- void cmd eraseprofile (usbdevice *kb, usbmode *dummy1, int dummy2, int dummy3, const char *dummy4)
- void freeprofile (usbdevice *kb)
- void hwtonative (usbprofile *profile, hwprofile *hw, int modecount)
- void nativetohw (usbprofile *profile, hwprofile *hw, int modecount)

Variables

```
static iconv_t utf8to16 = 0static iconv_t utf16to8 = 0
```

5.32.1 Function Documentation

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

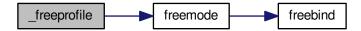
Definition at line 210 of file profile.c.

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

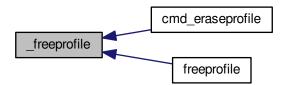
Referenced by cmd_eraseprofile(), and freeprofile().

```
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:



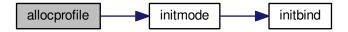
5.32.1.2 void allocprofile (usbdevice * kb)

Definition at line 182 of file profile.c.

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

Referenced by cmd_eraseprofile().

Here is the call graph for this function:



Here is the caller graph for this function:



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

Definition at line 203 of file profile.c.

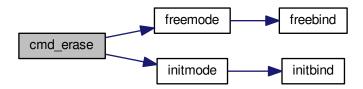
References freemode(), imutex, and initmode().

```
203
204 pthread_mutex_lock(imutex(kb));
205 freemode(mode);
206 initmode(mode);
207 pthread_mutex_unlock(imutex(kb));
208 }

{

{

    pthread_mutex_unlock(imutex(kb));
}
```



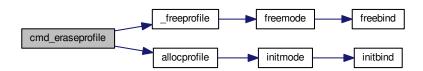
5.32.1.4 void cmd_eraseprofile (usbdevice * kb, usbmode * dummy1, int dummy2, int dummy3, const char * dummy4)

Definition at line 221 of file profile.c.

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

```
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:



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

Definition at line 160 of file profile.c.

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

```
160

161  // ID is either a GUID or an 8-digit hex number

162  int newmodified;

163  if(!setid(&mode->id, id) && sscanf(id, "%08x", &newmodified) == 1)

164  memcpy(mode->id.modified, &newmodified, sizeof(newmodified));

165 }
```

Here is the call graph for this function:

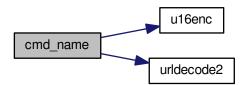


5.32.1.6 void cmd_name (usbdevice * kb, usbmode * mode, int dummy1, int dummy2, const char * name)

Definition at line 117 of file profile.c.

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

Here is the call graph for this function:



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

Definition at line 167 of file profile.c.

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

```
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 }
```

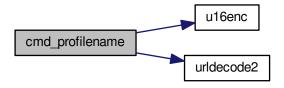


5.32.1.8 void cmd_profilename (usbdevice * kb, usbmode * dummy1, int dummy2, int dummy3, const char * name)

Definition at line 124 of file profile.c.

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

Here is the call graph for this function:



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

Definition at line 198 of file profile.c.

References usbmode::bind, and freebind().

Referenced by _freeprofile(), and cmd_erase().

Here is the call graph for this function:



Here is the caller graph for this function:

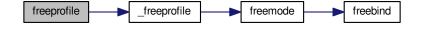


5.32.1.10 void freeprofile (usbdevice * kb)

Definition at line 228 of file profile.c.

References _freeprofile(), and usbdevice::hw.

Here is the call graph for this function:



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

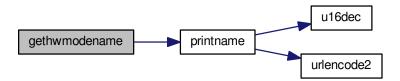
Definition at line 152 of file profile.c.

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

Referenced by _cmd_get().

```
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:



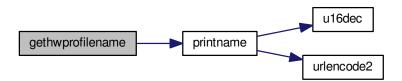
5.32.1.12 char* gethwprofilename (hwprofile * profile)

Definition at line 156 of file profile.c.

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

Referenced by _cmd_get().

Here is the call graph for this function:



Here is the caller graph for this function:



```
5.32.1.13 char* getid ( usbid * id )
```

Definition at line 79 of file profile.c.

References usbid::guid.

Referenced by _cmd_get().

```
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:



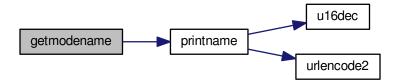
```
5.32.1.14 char* getmodename ( usbmode * mode )
```

Definition at line 144 of file profile.c.

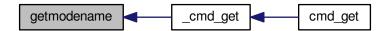
References MD NAME LEN, usbmode::name, and printname().

Referenced by _cmd_get().

```
144
145 return printname(mode->name, MD_NAME_LEN);
146 }
```



Here is the caller graph for this function:



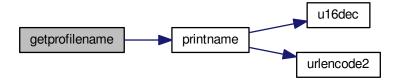
5.32.1.15 char* getprofilename (usbprofile * profile)

Definition at line 148 of file profile.c.

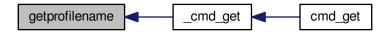
References usbprofile::name, PR_NAME_LEN, and printname().

Referenced by _cmd_get().

Here is the call graph for this function:



Here is the caller graph for this function:



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

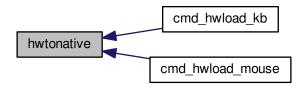
Definition at line 235 of file profile.c.

References usbmode::dpi, hwprofile::dpi, dpiset::forceupdate, lighting::forceupdate, usbmode::id, usbprofile::id, hwprofile::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:



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

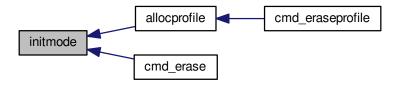
Definition at line 175 of file profile.c.

References usbmode::bind, usbmode::dpi, dpiset::forceupdate, lighting::forceupdate, initbind(), and usbmode::light. Referenced by allocprofile(), and cmd_erase().

```
175
176     memset(mode, 0, sizeof(*mode));
177     mode->light.forceupdate = 1;
178     mode->dpi.forceupdate = 1;
179     initbind(&mode->bind);
180 }
```



Here is the caller graph for this function:



5.32.1.18 int loadprofile (usbdevice * kb)

Definition at line 192 of file profile.c.

References hwloadprofile.

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

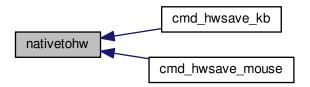
Definition at line 252 of file profile.c.

References usbmode::dpi, hwprofile::dpi, usbmode::id, usbprofile::id, hwprofile::id, usbmode::light, hwprofile::light, MD_NAME_LEN, usbprofile::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
```

Here is the caller graph for this function:



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

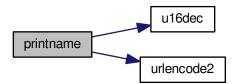
Definition at line 132 of file profile.c.

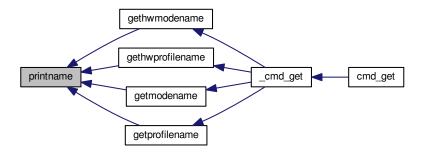
References u16dec(), and urlencode2().

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

```
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
           ul6dec(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:





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

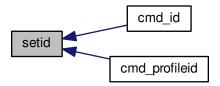
Definition at line 64 of file profile.c.

References usbid::guid.

Referenced by cmd_id(), and cmd_profileid().

```
64
65
                                      int32_t data1;
                                      int16_t data2, data3, data4a;
66
                                      char data4b[6];
                                     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%02hhX%02hhX%
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);
73
                                      memcpy(id->guid + 0x6, &data3, 2);
                                     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:



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

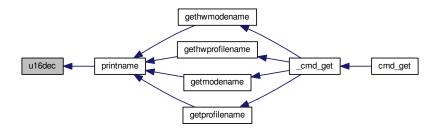
Definition at line 105 of file profile.c.

References utf16to8.

Referenced by printname().

```
105
106
1f(!utf16to8)
107
utf16to8 = iconv_open("UTF-8", "UTF-16LE");
108
size_t srclen2 = 0, srclenmax = *srclen;
109
for(; srclen2 < srclenmax; srclen2++) {
110
    if(!in[srclen2])
111
    break;
112
}
113
*srclen = srclen2 * 2;
114
iconv(utf16to8, (char**)&in, srclen, &out, dstlen);
115 }
```

Here is the caller graph for this function:



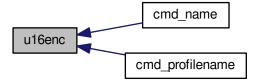
5.32.1.23 void u16enc (char * in, ushort * out, size_t * srclen, size_t * dstlen)

Definition at line 97 of file profile.c.

References utf8to16.

Referenced by cmd_name(), and cmd_profilename().

Here is the caller graph for this function:



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

Definition at line 8 of file profile.c.

Referenced by cmd_name(), and cmd_profilename().

```
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';

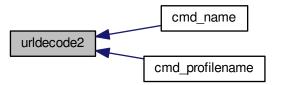
if(b >= 'a')

b -= 'a'-'A';

if(b >= 'A')

b -= 'A' - 10;
21
22
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:



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

Definition at line 37 of file profile.c.

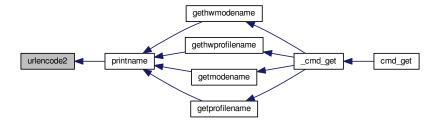
Referenced by printname().

```
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
50
                       b += 'A' - 10;
```

Here is the caller graph for this function:



5.32.2 Variable Documentation

```
5.32.2.1 iconv_t utf16to8 = 0 [static]
```

Definition at line 95 of file profile.c.

Referenced by u16dec().

```
5.32.2.2 iconv_t utf8to16 = 0 [static]
```

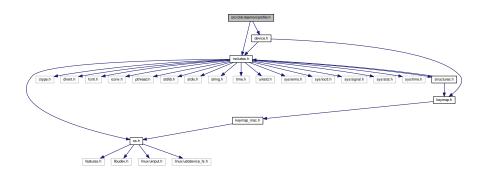
Definition at line 95 of file profile.c.

Referenced by u16enc().

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

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

Include dependency graph for profile.h:



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



Macros

• #define hwloadprofile(kb, apply) (kb)->vtable->hwload(kb, 0, 0, apply, 0)

Functions

- void allocprofile (usbdevice *kb)
- int loadprofile (usbdevice *kb)
- void freeprofile (usbdevice *kb)
- void cmd_erase (usbdevice *kb, usbmode *mode, int dummy1, int dummy2, const char *dummy3)
- void cmd_eraseprofile (usbdevice *kb, usbmode *dummy1, int dummy2, int dummy3, const char *dummy4)
- void cmd_name (usbdevice *kb, usbmode *mode, int dummy1, int dummy2, const char *name)
- void cmd_profilename (usbdevice *kb, usbmode *dummy1, int dummy2, int dummy3, const char *name)
- char * getmodename (usbmode *mode)
- char * getprofilename (usbprofile *profile)
- char * gethwmodename (hwprofile *profile, int index)
- char * gethwprofilename (hwprofile *profile)
- int setid (usbid *id, const char *guid)
- char * getid (usbid *id)
- void hwtonative (usbprofile *profile, hwprofile *hw, int modecount)
- void nativetohw (usbprofile *profile, hwprofile *hw, int modecount)
- void cmd_id (usbdevice *kb, usbmode *mode, int dummy1, int dummy2, const char *id)
- void cmd_profileid (usbdevice *kb, usbmode *mode, int dummy1, int dummy2, const char *id)
- int cmd_hwload_kb (usbdevice *kb, usbmode *dummy1, int dummy2, int apply, const char *dummy3)
- int cmd_hwload_mouse (usbdevice *kb, usbmode *dummy1, int dummy2, int apply, const char *dummy3)
- int cmd_hwsave_kb (usbdevice *kb, usbmode *dummy1, int dummy2, int dummy3, const char *dummy4)
- int cmd_hwsave_mouse (usbdevice *kb, usbmode *dummy1, int dummy2, int dummy3, const char *dummy4)

5.33.1 Macro Definition Documentation

5.33.1.1 #define hwloadprofile(kb, apply) (kb)->vtable->hwload(kb, 0, 0, apply, 0)

Definition at line 52 of file profile.h.

Referenced by _start_dev(), and loadprofile().

5.33.2 Function Documentation

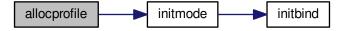
5.33.2.1 void allocprofile (usbdevice * kb)

Definition at line 182 of file profile.c.

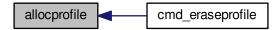
References usbprofile::currentmode, dpiset::forceupdate, lighting::forceupdate, initmode(), usbprofile::lastdpi, usbprofile::mode, MODE_COUNT, and usbdevice::profile.

Referenced by cmd_eraseprofile().

Here is the call graph for this function:



Here is the caller graph for this function:

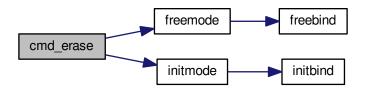


5.33.2.2 void cmd_erase (usbdevice * kb, usbmode * mode, int dummy1, int dummy2, const char * dummy3)

Definition at line 203 of file profile.c.

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

```
203
204 pthread_mutex_lock(imutex(kb));
205 freemode(mode);
206 initmode(mode);
207 pthread_mutex_unlock(imutex(kb));
208 }
```



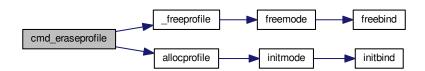
5.33.2.3 void cmd_eraseprofile (usbdevice * kb, usbmode * dummy1, int dummy2, int dummy3, const char * dummy4)

Definition at line 221 of file profile.c.

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

```
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:



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

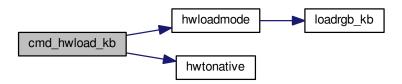
Definition at line 16 of file profile_keyboard.c.

References DELAY_LONG, usbdevice::hw, hwloadmode(), HWMODE_K70, HWMODE_K95, hwtonative(), hwprofile::id, IS_K95, MSG_SIZE, hwprofile::name, PR_NAME_LEN, usbdevice::profile, and usbrecv.

```
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
40
       // Load modes
       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)
       if(apply)
49
           hwtonative(kb->profile, hw, modes);
50
       // Free the existing profile (if any)
51
       free(kb->hw);
       kb->hw = hw;
52
53
       DELAY_LONG(kb);
       return 0;
55 }
```

Here is the call graph for this function:



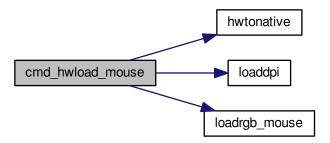
5.33.2.5 int cmd_hwload_mouse (usbdevice * kb, usbmode * dummy1, int dummy2, int apply, const char * dummy3)

Definition at line 6 of file profile_mouse.c.

References DELAY_LONG, hwprofile::dpi, usbdevice::hw, hwtonative(), hwprofile::id, hwprofile::light, loaddpi(), loadrgb_mouse(), MSG_SIZE, hwprofile::name, PR_NAME_LEN, usbdevice::profile, and usbrecv.

```
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);
31
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
      // Free the existing profile (if any)
44
      free(kb->hw);
      kb->hw = hw;
DELAY_LONG(kb);
45
46
      return 0;
47
48 }
```



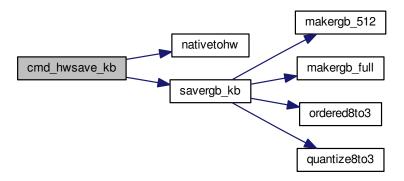
5.33.2.6 int cmd_hwsave_kb (usbdevice * kb, usbmode * dummy1, int dummy2, int dummy3, const char * dummy4)

Definition at line 57 of file profile_keyboard.c.

References DELAY_LONG, usbdevice::hw, HWMODE_K70, HWMODE_K95, hwprofile::id, IS_K95, hwprofile::light, MD_NAME_LEN, MSG_SIZE, hwprofile::name, nativetohw(), usbdevice::profile, savergb_kb(), and usbsend.

```
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
             if(!usbsend(kb, data_pkt[1], 1))
80
                 return -1;
81
```

Here is the call graph for this function:

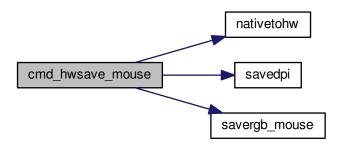


5.33.2.7 int cmd_hwsave_mouse (usbdevice * kb, usbmode * dummy1, int dummy2, int dummy3, const char * dummy4)

Definition at line 50 of file profile_mouse.c.

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

```
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 <= 1; i++) {</pre>
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;
77
       // Save the DPI data (also saves RGB for those states)
78
       if(savedpi(kb, hw->dpi, hw->light))
79
            return -1:
       DELAY_LONG(kb);
80
       return 0;
81
82 }
```



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

Definition at line 160 of file profile.c.

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

```
160

// ID is either a GUID or an 8-digit hex number

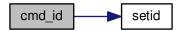
162 int newmodified;

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:

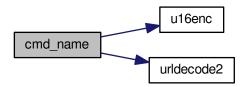


5.33.2.9 void cmd_name (usbdevice * kb, usbmode * mode, int dummy1, int dummy2, const char * name)

Definition at line 117 of file profile.c.

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

Here is the call graph for this function:



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

Definition at line 167 of file profile.c.

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

```
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 }
```

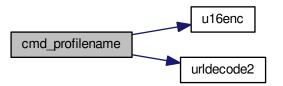
Here is the call graph for this function:



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

Definition at line 124 of file profile.c.

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



5.33.2.12 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:



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

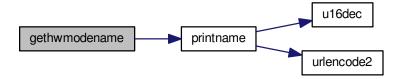
Definition at line 152 of file profile.c.

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

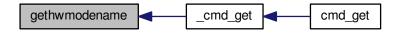
Referenced by _cmd_get().

```
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:



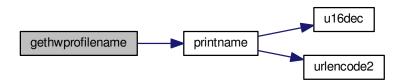
```
5.33.2.14 char* gethwprofilename ( hwprofile * profile )
```

Definition at line 156 of file profile.c.

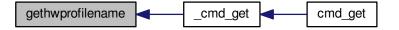
References MD_NAME_LEN, hwprofile::name, and printname().

Referenced by _cmd_get().

Here is the call graph for this function:



Here is the caller graph for this function:



```
5.33.2.15 char* getid ( usbid * id )
```

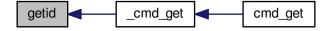
Definition at line 79 of file profile.c.

References usbid::guid.

Referenced by _cmd_get().

```
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:



```
5.33.2.16 char* getmodename ( usbmode * mode )
```

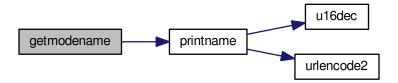
Definition at line 144 of file profile.c.

References MD NAME LEN, usbmode::name, and printname().

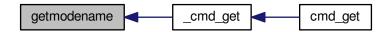
Referenced by _cmd_get().

```
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:



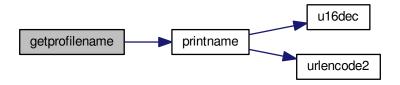
```
5.33.2.17 char* getprofilename ( usbprofile * profile )
```

Definition at line 148 of file profile.c.

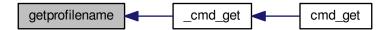
References usbprofile::name, PR_NAME_LEN, and printname().

Referenced by _cmd_get().

Here is the call graph for this function:



Here is the caller graph for this function:



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

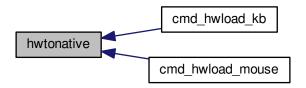
Definition at line 235 of file profile.c.

References usbmode::dpi, hwprofile::dpi, dpiset::forceupdate, lighting::forceupdate, usbmode::id, usbprofile::id, hwprofile::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:



5.33.2.19 int loadprofile (usbdevice * kb)

Definition at line 192 of file profile.c.

References hwloadprofile.

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

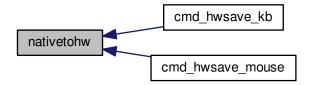
Definition at line 252 of file profile.c.

References usbmode::dpi, hwprofile::dpi, usbmode::id, usbprofile::id, hwprofile::id, usbmode::light, hwprofile::light, MD NAME LEN, usbprofile::mame, usbprofile::name, hwprofile::name, and PR NAME LEN.

Referenced by cmd_hwsave_kb(), and cmd_hwsave_mouse().

```
252
253
         // Copy name and ID
        memcpy(hw->name[0], profile->name, PR_NAME_LEN * 2);
254
255
        memcpy(hw->id, &profile->id, sizeof(usbid));
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);
260
             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));
2.61
262
263
         }
264 }
```

Here is the caller graph for this function:



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

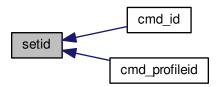
Definition at line 64 of file profile.c.

References usbid::guid.

Referenced by cmd_id(), and cmd_profileid().

```
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
                                                                                                 \& data1, \& data2, \& data3, \& data4a, data4b, data4b + 1, data4b + 2, data4b + 3, data4b + 4, \\
69
                                  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;
```

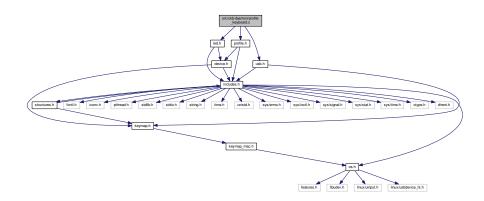
Here is the caller graph for this function:



5.34 src/ckb-daemon/profile_keyboard.c File Reference

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

Include dependency graph for profile_keyboard.c:



Functions

- static int hwloadmode (usbdevice *kb, hwprofile *hw, int mode)
- int cmd_hwload_kb (usbdevice *kb, usbmode *dummy1, int dummy2, int apply, const char *dummy3)
- int cmd_hwsave_kb (usbdevice *kb, usbmode *dummy1, int dummy2, int dummy3, const char *dummy4)

5.34.1 Function Documentation

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

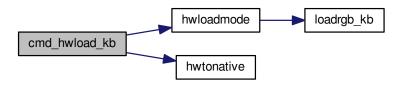
Definition at line 16 of file profile_keyboard.c.

References DELAY_LONG, usbdevice::hw, hwloadmode(), HWMODE_K70, HWMODE_K95, hwtonative(), hwprofile::id, IS K95, MSG SIZE, hwprofile::name, PR NAME LEN, usbdevice::profile, and usbrecv.

```
16
17 DELAY_LONG(kb);
18 hwprofile* hw = calloc(1, sizeof(hwprofile));
19 // Ask for profile and mode IDs
```

```
uchar data_pkt[2][MSG_SIZE] = {
20
            { 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);
54
        return 0;
55 }
```

Here is the call graph for this function:



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

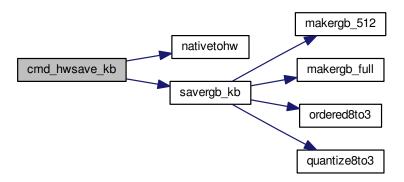
Definition at line 57 of file profile_keyboard.c.

References DELAY_LONG, usbdevice::hw, HWMODE_K70, HWMODE_K95, hwprofile::id, IS_K95, hwprofile::light, MD_NAME_LEN, MSG_SIZE, hwprofile::name, nativetohw(), usbdevice::profile, savergb_kb(), and usbsend.

```
57
        DELAY_LONG(kb);
58
        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);
73
          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:



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

Definition at line 5 of file profile_keyboard.c.

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

Referenced by cmd_hwload_kb().

Here is the call graph for this function:



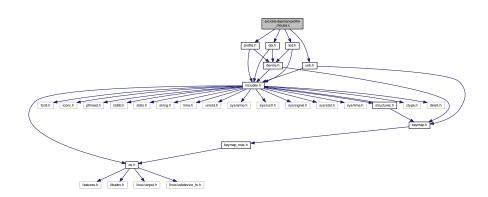
Here is the caller graph for this function:



5.35 src/ckb-daemon/profile_mouse.c File Reference

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

Include dependency graph for profile_mouse.c:



Functions

- int cmd_hwload_mouse (usbdevice *kb, usbmode *dummy1, int dummy2, int apply, const char *dummy3)
- int cmd_hwsave_mouse (usbdevice *kb, usbmode *dummy1, int dummy2, int dummy3, const char *dummy4)

5.35.1 Function Documentation

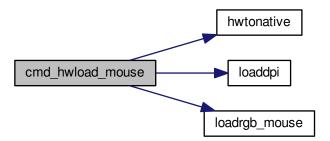
5.35.1.1 int cmd_hwload_mouse (usbdevice * kb, usbmode * dummy1, int dummy2, int apply, const char * dummy3)

Definition at line 6 of file profile mouse.c.

References DELAY_LONG, hwprofile::dpi, usbdevice::hw, hwtonative(), hwprofile::id, hwprofile::light, loaddpi(), loadrgb_mouse(), MSG_SIZE, hwprofile::name, PR_NAME_LEN, usbdevice::profile, and usbrecv.

```
7
      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
14
       uchar in_pkt[MSG_SIZE];
       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++) {</pre>
            data_pkt[1][3] = i;
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
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:



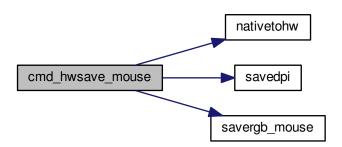
5.35.1.2 int cmd_hwsave_mouse (usbdevice * kb, usbmode * dummy1, int dummy2, int dummy3, const char * dummy4)

Definition at line 50 of file profile_mouse.c.

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

```
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 }
```

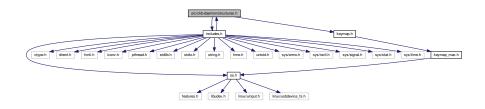
Here is the call graph for this function:



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

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

Include dependency graph for structures.h:



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



Data Structures

- struct usbid
- · struct macroaction
- · struct keymacro
- · struct binding
- struct dpiset
- · struct lighting
- struct usbmode
- struct usbprofile
- · struct hwprofile
- struct usbinput
- struct usbdevice

Macros

- #define SET_KEYBIT(array, index) do { (array)[(index) / 8] |= 1 << ((index) % 8); } while(0)
- #define CLEAR_KEYBIT(array, index) do { (array)[(index) / 8] &= ~(1 << ((index) % 8)); } while(0)
- #define I_NUM 1
- #define I_CAPS 2
- #define I SCROLL 4
- #define OUTFIFO_MAX 10
- #define MACRO_MAX 1024
- #define DPI_COUNT 6
- #define LIFT_MIN 1
- #define LIFT_MAX 5
- #define MD_NAME_LEN 16
- #define PR_NAME_LEN 16
- #define MODE_COUNT 6
- #define HWMODE_K70 1
- #define HWMODE_K95 3
- #define HWMODE_MAX 3
- #define FEAT_RGB 0x001
- #define FEAT MONOCHROME 0x002
- #define FEAT_POLLRATE 0x004

- #define FEAT_ADJRATE 0x008
- #define FEAT_BIND 0x010
- #define FEAT NOTIFY 0x020
- #define FEAT_FWVERSION 0x040
- #define FEAT_FWUPDATE 0x080
- #define FEAT_HWLOAD 0x100
- #define FEAT ANSI 0x200
- #define FEAT_ISO 0x400
- #define FEAT MOUSEACCEL 0x800
- #define FEAT_COMMON (FEAT_BIND | FEAT_NOTIFY | FEAT_FWVERSION | FEAT_MOUSEACCEL | FEAT_HWLOAD)
- #define FEAT_STD_RGB (FEAT_COMMON | FEAT_RGB | FEAT_POLLRATE | FEAT_FWUPDATE)
- #define FEAT_STD_NRGB (FEAT_COMMON)
- #define FEAT_LMASK (FEAT_ANSI | FEAT_ISO)
- #define HAS FEATURES(kb, feat) (((kb)->features & (feat)) == (feat))
- #define HAS ANY FEATURE(kb, feat) (!!((kb)->features & (feat)))
- #define NEEDS_FW_UPDATE(kb) ((kb)->fwversion == 0 && HAS_FEATURES((kb), FEAT_FWUPDATE |
 FEAT_FWVERSION))
- #define SCROLL_ACCELERATED 0
- #define SCROLL MIN 1
- #define SCROLL MAX 10
- #define KB_NAME_LEN 34
- #define SERIAL_LEN 34
- #define MSG_SIZE 64
- #define IFACE_MAX 4

Variables

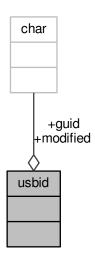
- · const union devcmd vtable_keyboard
- const union devcmd vtable keyboard nonrgb
- · const union devcmd vtable_mouse

5.36.1 Data Structure Documentation

5.36.1.1 struct usbid

Definition at line 8 of file structures.h.

Collaboration diagram for usbid:



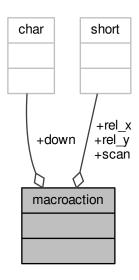
Data Fields

char	guid[16]	
char	modified[4]	

5.36.1.2 struct macroaction

Definition at line 27 of file structures.h.

Collaboration diagram for macroaction:



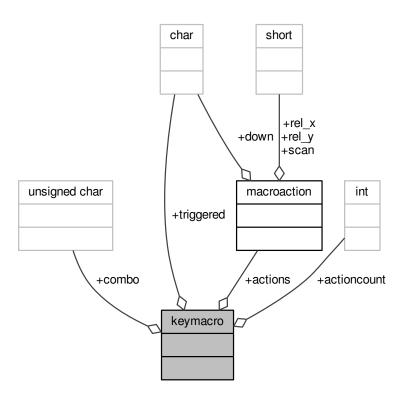
Data Fields

char	down	
short	rel_x	
short	rel_y	
short	scan	

5.36.1.3 struct keymacro

Definition at line 34 of file structures.h.

Collaboration diagram for keymacro:



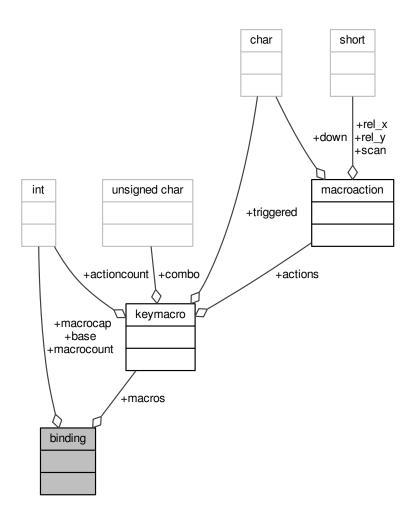
Data Fields

int	actioncount	
macroaction *	actions	
uchar	combo[((((152+3+	2)+25)+7)/8)]
char	triggered	

5.36.1.4 struct binding

Definition at line 42 of file structures.h.

Collaboration diagram for binding:



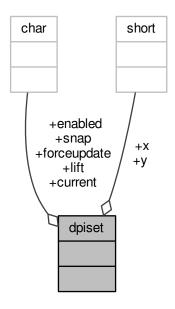
Data Fields

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

5.36.1.5 struct dpiset

Definition at line 56 of file structures.h.

Collaboration diagram for dpiset:



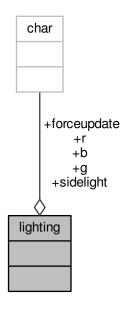
Data Fields

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

5.36.1.6 struct lighting

Definition at line 72 of file structures.h.

Collaboration diagram for lighting:



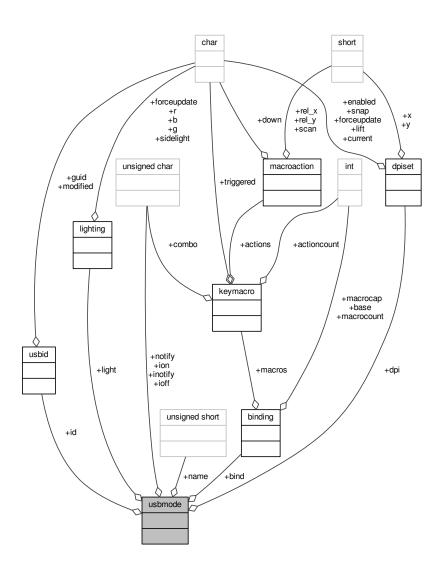
Data Fields

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

5.36.1.7 struct usbmode

Definition at line 82 of file structures.h.

Collaboration diagram for usbmode:



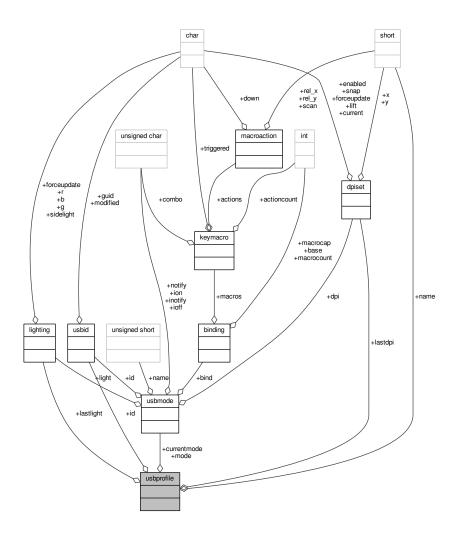
Data Fields

binding	bind
dpiset	dpi
usbid	id
uchar	inotify[10]
uchar	ioff
uchar	ion
lighting	light
ushort	name[16]
uchar	notify[10][((((152+3+12)+25)+7)/8)]

5.36.1.8 struct usbprofile

Definition at line 100 of file structures.h.

Collaboration diagram for usbprofile:



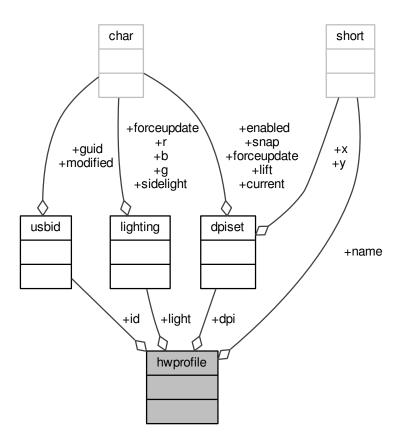
Data Fields

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

5.36.1.9 struct hwprofile

Definition at line 117 of file structures.h.

Collaboration diagram for hwprofile:



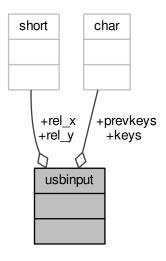
Data Fields

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

5.36.1.10 struct usbinput

Definition at line 128 of file structures.h.

Collaboration diagram for usbinput:



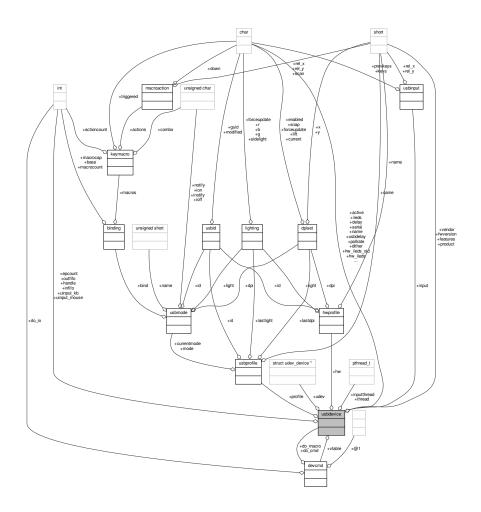
Data Fields

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

5.36.1.11 struct usbdevice

Definition at line 177 of file structures.h.

Collaboration diagram for usbdevice:



Data Fields

char	active	
char	delay	
char	dither	
int	epcount	
ushort	features	
ushort	fwversion	
int	handle	
hwprofile *	hw	
uchar	hw_ileds	
uchar	hw_ileds_old	
uchar	ileds	
int	infifo	
usbinput	input	
pthread_t	inputthread	
char	name[34+1]	

int	outfifo[10]	
char	pollrate	
short	product	
usbprofile *	profile	
char	serial[34]	
pthread_t	thread	
struct	udev	
udev_device *		
int	uinput_kb	
int	uinput_mouse	
char	usbdelay	
short	vendor	
const union	vtable	
devcmd *		

5.36.2 Macro Definition Documentation

5.36.2.1 #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().

5.36.2.2 #define DPI_COUNT 6

Definition at line 53 of file structures.h.

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

5.36.2.3 #define FEAT_ADJRATE 0x008

Definition at line 138 of file structures.h.

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

5.36.2.4 #define FEAT_ANSI 0x200

Definition at line 145 of file structures.h.

Referenced by readcmd().

5.36.2.5 #define FEAT_BIND 0x010

Definition at line 139 of file structures.h.

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

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

Definition at line 150 of file structures.h.

5.36.2.7 #define FEAT_FWUPDATE 0x080

Definition at line 142 of file structures.h.

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

5.36.2.8 #define FEAT_FWVERSION 0x040

Definition at line 141 of file structures.h.

Referenced by _mkdevpath(), and _start_dev().

5.36.2.9 #define FEAT_HWLOAD 0x100

Definition at line 143 of file structures.h.

Referenced by _start_dev().

5.36.2.10 #define FEAT_ISO 0x400

Definition at line 146 of file structures.h.

Referenced by readcmd().

5.36.2.11 #define FEAT_LMASK (FEAT_ANSI | FEAT_ISO)

Definition at line 153 of file structures.h.

Referenced by readcmd().

5.36.2.12 #define FEAT_MONOCHROME 0x002

Definition at line 136 of file structures.h.

Referenced by _mkdevpath(), and _setupusb().

5.36.2.13 #define FEAT_MOUSEACCEL 0x800

Definition at line 147 of file structures.h.

Referenced by main(), and readcmd().

5.36.2.14 #define FEAT_NOTIFY 0x020

Definition at line 140 of file structures.h.

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

5.36.2.15 #define FEAT_POLLRATE 0x004

Definition at line 137 of file structures.h.

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

5.36.2.16 #define FEAT_RGB 0x001

Definition at line 135 of file structures.h.

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

5.36.2.17 #define FEAT_STD_NRGB (FEAT_COMMON)

Definition at line 152 of file structures.h.

Referenced by _setupusb().

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

Definition at line 151 of file structures.h.

Referenced by _setupusb().

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

Definition at line 157 of file structures.h.

5.36.2.20 #define HAS_FEATURES(kb, feat) (((kb)->features & (feat)) == (feat))

Definition at line 156 of file structures.h.

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

5.36.2.21 #define HWMODE_K70 1

Definition at line 114 of file structures.h.

Referenced by cmd_hwload_kb(), and cmd_hwsave_kb().

5.36.2.22 #define HWMODE_K95 3

Definition at line 115 of file structures.h.

Referenced by cmd_hwload_kb(), and cmd_hwsave_kb().

5.36.2.23 #define HWMODE_MAX 3

Definition at line 116 of file structures.h.

5.36.2.24 #define I_CAPS 2

Definition at line 20 of file structures.h.

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

5.36.2.25 #define I_NUM 1

Definition at line 19 of file structures.h.

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

5.36.2.26 #define I_SCROLL 4

Definition at line 21 of file structures.h.

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

5.36.2.27 #define IFACE_MAX 4

Definition at line 176 of file structures.h.

5.36.2.28 #define KB_NAME_LEN 34

Definition at line 173 of file structures.h.

Referenced by _setupusb(), and os_setupusb().

5.36.2.29 #define LIFT_MAX 5

Definition at line 55 of file structures.h.

Referenced by cmd_lift(), and loaddpi().

5.36.2.30 #define LIFT_MIN 1

Definition at line 54 of file structures.h.

Referenced by cmd_lift(), and loaddpi().

5.36.2.31 #define MACRO_MAX 1024

Definition at line 50 of file structures.h.

Referenced by _cmd_macro().

5.36.2.32 #define MD_NAME_LEN 16

Definition at line 81 of file structures.h.

Referenced by cmd_hwsave_kb(), cmd_hwsave_mouse(), cmd_name(), gethwmodename(), gethwprofilename(), gethwdename(), hwloadmode(), hwtonative(), and nativetohw().

5.36.2.33 #define MODE_COUNT 6

Definition at line 99 of file structures.h.

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

5.36.2.34 #define MSG_SIZE 64

Definition at line 175 of file structures.h.

Referenced by _usbsend(), cmd_hwload_kb(), cmd_hwload_mouse(), cmd_hwsave_kb(), cmd_hwsave_mouse(), cmd_pollrate(), fwupdate(), getfwversion(), hwloadmode(), loaddpi(), loadrgb_kb(), loadrgb_mouse(), os_inputmain(), os_usbrecv(), os_usbsend(), savedpi(), savergb_kb(), savergb_mouse(), setactive_kb(), setactive_mouse(), updatedpi(), updatergb_kb(), and updatergb_mouse().

5.36.2.35 #define NEEDS_FW_UPDATE(kb) ((kb)->fwversion == 0 && HAS_FEATURES((kb), FEAT_FWUPDATE | FEAT_FWVERSION))

Definition at line 160 of file structures.h.

Referenced by _start_dev(), readcmd(), revertusb(), setactive_kb(), and setactive_mouse().

5.36.2.36 #define OUTFIFO_MAX 10

Definition at line 24 of file structures.h.

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

5.36.2.37 #define PR_NAME_LEN 16

Definition at line 98 of file structures.h.

Referenced by cmd_hwload_kb(), cmd_hwload_mouse(), cmd_profilename(), getprofilename(), hwtonative(), and nativetohw().

5.36.2.38 #define SCROLL_ACCELERATED 0

Definition at line 163 of file structures.h.

Referenced by readcmd().

5.36.2.39 #define SCROLL MAX 10

Definition at line 165 of file structures.h.

Referenced by readcmd().

5.36.2.40 #define SCROLL_MIN 1

Definition at line 164 of file structures.h.

Referenced by readcmd().

5.36.2.41 #define SERIAL_LEN 34

Definition at line 174 of file structures.h.

Referenced by _setupusb(), and os_setupusb().

5.36.2.42 #define SET_KEYBIT(array, index) do { (array)[(index) / 8] |= 1 << ((index) % 8); } while(0)

Definition at line 15 of file structures.h.

Referenced by _cmd_macro(), cmd_notify(), corsair_mousecopy(), hid_kb_translate(), and hid_mouse_translate().

5.36.3 Variable Documentation

5.36.3.1 const union devcmd vtable_keyboard

Definition at line 28 of file device vtable.c.

Referenced by get_vtable().

5.36.3.2 const union devcmd vtable_keyboard_nonrgb

Definition at line 75 of file device_vtable.c.

Referenced by get_vtable().

5.36.3.3 const union devcmd vtable_mouse

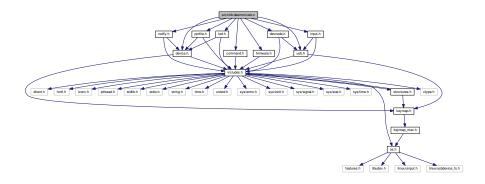
Definition at line 122 of file device_vtable.c.

Referenced by get_vtable().

5.37 src/ckb-daemon/usb.c File Reference

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

Include dependency graph for usb.c:



Functions

```
• const char * vendor_str (short vendor)
```

brief .

• const char * product_str (short product)

brief

• static const devcmd * get_vtable (short vendor, short product)

brief.

static void * devmain (usbdevice *kb)

brief

static void * _setupusb (void *context)

brief

- void setupusb (usbdevice *kb)
- int revertusb (usbdevice *kb)
- int _resetusb (usbdevice *kb, const char *file, int line)
- int usb_tryreset (usbdevice *kb)
- int <u>usbsend</u> (usbdevice *kb, const uchar *messages, int count, const char *file, int line)
- int _usbrecv (usbdevice *kb, const uchar *out_msg, uchar *in_msg, const char *file, int line)
- int closeusb (usbdevice *kb)

Variables

```
    pthread_mutex_t usbmutex = PTHREAD_MUTEX_INITIALIZER brief.
    volatile int reset_stop = 0 brief.
    int features_mask = -1 brief.
    int hwload_mode
```

5.37.1 Function Documentation

```
5.37.1.1 int resetusb ( usbdevice * kb, const char * file, int line )
```

resetusb Reset a USB device.

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

Definition at line 426 of file usb.c.

References usbdevice::active, DELAY LONG, os resetusb(), and usbdevice::vtable.

```
427
        // Perform a USB reset
428
        DELAY_LONG(kb);
429
        int res = os_resetusb(kb, file, line);
430
        if (res)
431
             return res;
        DELAY_LONG(kb);
432
433
        // Re-initialize the device
        if(kb->vtable->start(kb, kb->active) != 0)
    return -1;
434
435
436
        if (kb->vtable->updatergb(kb, 1) != 0)
437
            return -1;
438
        return 0;
439 }
```

Here is the call graph for this function:



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

setupusb A horrible function for setting up an usb device

Parameters

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

Returns

a ptread t* 0, here casted as void*. Retval is always null

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

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

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

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

The basic idea of this routine is the following:

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

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

• The same happens with os_setupindicators(), which initially initializes all LED variables in kb to off and then starts the _ledthread() thread with kb as parameter and then detaches it. Here again only the generation of the thread can fail.

• Via an entry in the vable (allocprofile, identical for all three vtable types), allocprofile() is called in profile.c. With a valid parameter kb, a usbprofile structure is allocated and stored as a kb->profile. Then initmode() is called for each of the initializable modes (MODE_COUNT, currently 6). This procedure creates the memory space for the mode information, initializes the range to 0, and then sets the light.forceupdate and dpi.forceupdate to true. This forces an update later in the initialization of the device.

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

Warning

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

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

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

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

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

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

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

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

Then two dummy values (active and pollrate) are set in the kb structure and ready.

- start_dev() does a bit more because this function is for both mouse and keyboard. start_dev() calls after setting an extended timeout parameter start dev(). Both are located in device.c.
- First, _start_dev() attempts to determine the firmware version of the device, but only if two conditions
 are met: hwload-mode is not null (then hw-loading is disabled) and the device has the FEAT_HWLOAD
 feature. Then the firmware and the poll rate are fetched via getfwersion().
 - If hwload_mode is set to "load only once" (==1), then the HWLOAD feature is masked, so that no further reading can take place.
- Now check if device needs a firmware update. If so, set it up and leave the function without error.
- Else load the hardware profile from device if the hw-pointer is not set and hw-loading is possible and allowed.
 - Return error if mode == 2 (load always) and loading got an error. Else mask the HWLOAD feature, because hwload must be 1 and the error could be a repeated hw-reading.

Puh, that is real Horror code. It seems to be not faulty, but completely unreadable.

Finally, the second parameter of _startdev() is used to check whether the device is to be activated.
 Depending on the parameter, the active or the idle-member in the correspondig vtable is called. These are device-dependent again:

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

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

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

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

Definition at line 214 of file usb.c.

References ckb_info, closeusb(), DELAY_LONG, devmain(), devpath, dmutex, FEAT_ADJRATE, FEAT_MONOCHROME, FEAT_STD_NRGB, FEAT_STD_RGB, usbdevice::features, features_mask, get_vtable(), imutex, INDEX_OF, usbdevice::inputthread, IS_MONOCHROME, IS_MOUSE, IS_RGB, KB_NAME_LEN, keyboard, mkdevpath(), usbdevice::name, os_inputmain(), os_inputopen(), os_setupindicators(), os_setupusb(), usbdevice::product, product_str(), usbdevice::serial, SERIAL_LEN, updateconnected(), USB_DELAY_DEFAULT, usb_tryreset(), usbdevice::usbdelay, usbdevice::vendor, vendor_str(), and usbdevice::vtable.

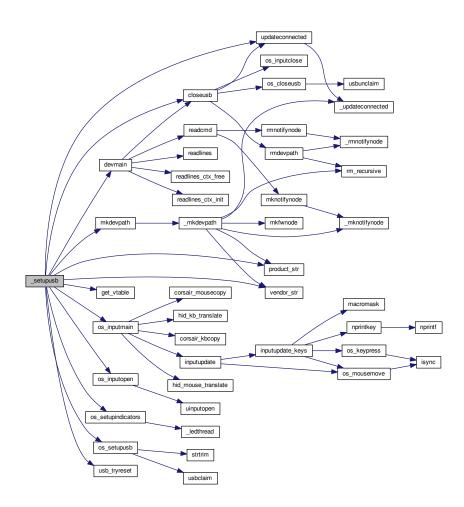
Referenced by setupusb().

```
214
227
        usbdevice* kb = context;
228
        // Set standard fields
        short vendor = kb->vendor, product = kb->product;
229
        const devcmd* vt = kb->vtable = get_vtable(vendor, product);
230
        kb->features = (IS_RGB(vendor, product) ? FEAT_STD_RGB :
      FEAT_STD_NRGB) & features_mask;
        if(IS_MOUSE(vendor, product)) kb->features |= FEAT_ADJRATE;
232
233
        if(IS_MONOCHROME(vendor, product)) kb->features |=
      FEAT MONOCHROME;
234
        kb->usbdelay = USB_DELAY_DEFAULT;
235
236
        // Perform OS-specific setup
240
       DELAY_LONG(kb);
241
        if (os setupusb(kb))
247
248
            goto fail:
249
255
        // Make up a device name and serial if they weren't assigned
256
257
            snprintf(kb->serial, SERIAL_LEN, "%04x:%04x-NoID", kb->
      vendor, kb->product);
258
       if(!kb->name[0])
            snprintf(kb->name, KB_NAME_LEN, "%s %s", vendor_str(kb->
259
      vendor), product_str(kb->product));
260
261
        // Set up an input device for key events
269
        if (os_inputopen(kb))
270
            goto fail;
        if(pthread_create(&kb->inputthread, 0, os_inputmain, kb))
275
            goto fail;
        pthread_detach(kb->inputthread);
276
282
        if (os_setupindicators(kb))
283
            goto fail;
284
        // Set up device
285
        vt->allocprofile(kb);
```

```
309
         vt->updateindicators(kb, 1);
         pthread_mutex_unlock(imutex(kb));
if(vt->start(kb, 0) && usb_tryreset(kb))
314
348
         goto fail_noinput;

// Make /dev path
if (mkdevpath (kb))
goto fail_noinput;
349
355
356
357
363
          // Finished. Enter main loop
          int index = INDEX_OF(kb, keyboard);
364
         ckb_info("Setup finished for %s%d\n", devpath, index);
365
         updateconnected();
366
369
         return devmain(kb);
372
         fail:
373
         pthread_mutex_unlock(imutex(kb));
374
         fail_noinput:
375
376
         closeusb(kb);
         pthread_mutex_unlock(dmutex(kb));
377
          return 0;
378 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



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

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

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

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

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

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

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

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

os usbrecv() returns 0, -1 or something else.

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

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

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

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

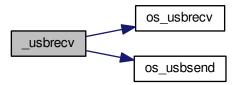
The buffers behind **out_msg** and **in_msg** are MSG_SIZE at least (currently 64 Bytes). More is ok but useless, less brings unpredictable behavior.

Definition at line 599 of file usb.c.

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

```
599
600
        // Try a maximum of 3 times \,
        for(int try = 0; try < 5; try++) {</pre>
601
602
            // Send the output message
            DELAY_SHORT (kb);
604
            int res = os_usbsend(kb, out_msg, 1, file, line);
            if(res == 0)
605
606
                return 0;
            else if (res == -1) {
607
                // Retry on temporary failure
608
                if(reset_stop)
610
611
                DELAY_LONG(kb);
612
                continue;
613
            // Wait for the response
614
            DELAY_MEDIUM(kb);
615
            res = os_usbrecv(kb, in_msg, file, line);
617
            if(res == 0)
618
                return 0:
619
            else if (res != -1)
620
                return res;
621
            if(reset_stop || hwload_mode != 2)
622
            DELAY_LONG(kb);
623
624
        // Give up
62.5
626
        ckb_err_fn("Too many send/recv failures. Dropping.\n", file, line);
627
        return 0;
628 }
```

Here is the call graph for this function:



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

_usbsend send a logical message completely to the given device

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

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

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

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

Attention

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

An essential constant parameter which is relevant for os_usbsend() only is is_recv = 0, which means sending. Now it gets a little complicated again:

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

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

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

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

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

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

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

Definition at line 532 of file usb.c.

References DELAY LONG, DELAY SHORT, hwload mode, MSG SIZE, os usbsend(), and reset stop.

```
532
533
        int total_sent = 0;
534
        for(int i = 0; i < count; i++) {</pre>
             // Send each message via the OS function
535
536
             while(1){
                 DELAY_SHORT (kb);
537
538
                 int res = os_usbsend(kb, messages + i * MSG_SIZE, 0, file, line);
                 if(res == 0)
                     return 0;
540
541
                 else if (res !=-1) {
542
                     total_sent += res;
543
                     break:
544
545
                 . // Stop immediately if the program is shutting down or hardware load is set to tryonce
546
                 if(reset_stop || hwload_mode != 2)
                 return 0;
// Retry as long as the result is temporary failure
547
548
                 DELAY LONG(kb);
549
550
551
        return total_sent;
553 }
```

Here is the call graph for this function:



5.37.1.5 int closeusb (usbdevice *kb)

closeusb Close a USB device and remove device entry.

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

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

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

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

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

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

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

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

See Also

the hack note with rmdevpath())

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

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

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

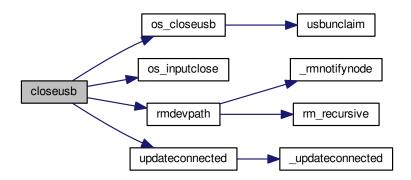
Definition at line 673 of file usb.c.

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

Referenced by setupusb(), devmain(), quitWithLock(), and usb rm device().

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

Here is the call graph for this function:



Here is the caller graph for this function:



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

devmain is called by _setupusb

Parameters

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

Returns

always a nullptr

Synchronization

The syncing via mutexes is interesting:

1. imutex (the Input mutex)

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

Warning

have a look at updateindicators() later.

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

2. dmutex (the Device mutex)

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

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

Todo Hope to find the need for dmutex usage later.

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

Attention

dmutex should still be locked when this is called

First a readlines ctx buffer structure is initialized by readlines_ctx_init().

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

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

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

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

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

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

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

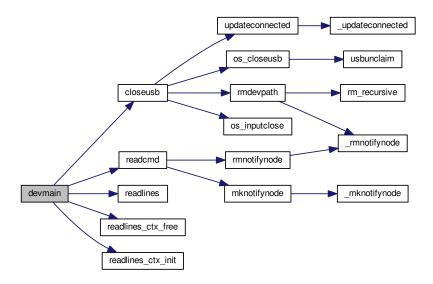
Definition at line 135 of file usb.c.

References closeusb(), dmutex, usbdevice::infifo, IS_CONNECTED, readcmd(), readlines(), readlines_ctx_free(), and readlines_ctx_init().

Referenced by setupusb().

```
135
137
        int kbfifo = kb->infifo - 1;
        readlines_ctx linectx;
140
141
        readlines ctx init(&linectx);
146
        while(1){
            pthread_mutex_unlock(dmutex(kb));
154
            // Read from FIFO
155
            const char* line:
            int lines = readlines(kbfifo, linectx, &line);
156
            pthread_mutex_lock(dmutex(kb));
157
158
             // End thread when the handle is removed
            if(!IS_CONNECTED(kb))
159
160
            if(lines) {
164
                 if(readcmd(kb, line)){
167
173
                     // USB transfer failed; destroy device
174
                     closeusb(kb);
175
                     break;
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.37.1.7 static const devcmd* get_vtable (short *vendor*, short *product*) [static]

get_vtable returns the correct vtable pointer

Parameters

vendor	short usb vendor ID
product	short usb product ID

Returns

Depending on the type and model, the corresponding vtable pointer is returned (see below)

At present, we have three different vtables:

- vtable_mouse is used for all mouse types. This may be wrong with some newer mice?
- vtable_keyboard is used for all RGB Keyboards.
- \bullet vtable_keyboard_nonrgb for all the rest.

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

Definition at line 102 of file usb.c.

References IS_MOUSE, IS_RGB, vtable_keyboard, vtable_keyboard_nonrgb, and vtable_mouse.

Referenced by _setupusb().

```
102

103

return IS_MOUSE(vendor, product) ? &vtable_mouse :
    IS_RGB(vendor, product) ? &vtable_keyboard : &
    vtable_keyboard_nonrgb;

104 }
```

Here is the caller graph for this function:



5.37.1.8 const char* product_str (short product)

product_str returns a condensed view on what type of device we have.

At present, various models and their properties are known from corsair products. Some models differ in principle (mice and keyboards), others differ in the way they function (for example, RGB and non RGB), but they are very similar.

Here, only the first point is taken into consideration and we return a unified model string. If the model is not known with its number, *product_str* returns an empty string.

The model numbers and corresponding strings wwith the numbers in hex-string are defined in usb.h

At present, this function is used to initialize kb->name and to give information in debug strings.

Attention

The combinations below have to fit to the combinations in the macros mentioned above. So if you add a device with a new number, change both.

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

Definition at line 70 of file usb.c.

References P_HARPOON, P_K65, P_K65_LUX, P_K65_NRGB, P_K65_RFIRE, P_K70, P_K70_LUX, P_K70_LUX_NRGB, P_K70_NRGB, P_K70_RFIRE, P_K70_RFIRE_NRGB, P_K95, P_K95_NRGB, P_K95_PLATINUM, P_M65, P_M65_PRO, P_SABRE_L, P_SABRE_N, P_SABRE_O, P_SABRE_O2, P_SCIMITAR, P_SCIMITAR_P-RO, P_STRAFE_NRGB.

Referenced by _mkdevpath(), and _setupusb().

```
70
       if(product == P_K95 || product == P_K95_NRGB || product ==
71
      P_K95_PLATINUM)
           return "k95";
72
       if(product == P_K70 || product == P_K70_NRGB || product ==
      P_K70_LUX || product == P_K70_LUX_NRGB || product =
      P_K70_RFIRE || product == P_K70_RFIRE_NRGB)
74
           return "k70";
       if (product == P_K65 || product == P_K65_NRGB || product ==
75
      P_K65_LUX || product == P_K65_RFIRE)
return "k65";
76
       if(product == P_STRAFE || product == P_STRAFE_NRGB)
```

```
78          return "strafe";
79          if(product == P_M65 || product == P_M65_PRO)
80                return "m65";
81          if(product == P_SABRE_O || product == P_SABRE_L || product == P_SABRE_N || product == P_SABRE_O2 || product == P_HARPOON)
82          return "sabre";
83          if(product == P_SCIMITAR || product == P_SCIMITAR_PRO)
84          return "scimitar";
85          return ";
86 }
```

Here is the caller graph for this function:



5.37.1.9 int revertusb (usbdevice * kb)

revertusb sets a given device to inactive (hardware controlled) mode if not a fw-ugrade is indicated First is checked, whether a firmware-upgrade is indicated for the device. If so, revertusb() returns 0.

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

Anyway, the following steps are similar to some other procs, dealing with low level usb handling:

• If we do not have an RGB device, a simple setting to Hardware-mode (NK95_HWON) is sent to the device via n95cmd().

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

• If we have an RGB device, setactive() is called with second param active = false. That function will have a look on differences between keyboards and mice.

More precisely setactive() is just a macro to call via the kb->vtable enties either the active() or the idle() function where the vtable points to. setactive() may return error indications. If so, revertusb() returns -1, otherwise 0 in any other case.

Definition at line 407 of file usb.c.

References FEAT_RGB, HAS_FEATURES, NEEDS_FW_UPDATE, NK95_HWON, nk95cmd, and setactive.

```
Referenced by quitWithLock().
```

```
407
        if (NEEDS_FW_UPDATE(kb))
408
409
            return 0;
        if(!HAS_FEATURES(kb, FEAT_RGB)){
410
411
            nk95cmd(kb, NK95_HWON);
412
            return 0;
413
        if(setactive(kb, 0))
414
415
            return -1:
416
        return 0;
417 }
```

Here is the caller graph for this function:



5.37.1.10 void setupusb (usbdevice * kb)

setupusb starts a thread with kb as parameter and _setupusb() as entrypoint.

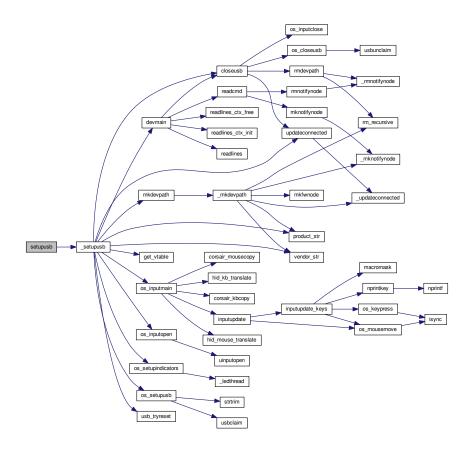
Set up a USB device after its handle is open. Spawns a new thread _setupusb() with standard parameter kb. dmutex must be locked prior to calling this function. The function will unlock it when finished. In kb->thread the thread id is mentioned, because closeusb() needs this info for joining that thread again.

Definition at line 386 of file usb.c.

References setupusb(), ckb err, imutex, and usbdevice::thread.

Referenced by usbadd().

Here is the call graph for this function:



Here is the caller graph for this function:



```
5.37.1.11 int usb_tryreset ( usbdevice * kb )
```

usb_tryreset does what the name means: Try to reset the usb via resetusb()

This function is called if an usb command ran into an error in case of one of the following two situations:

When setting up a new usb device and the start() function got an error (
 See Also

```
_setupusb())
```

· If upgrading to a new firmware gets an error (

See Also

```
cmd_fwupdate()).
```

The previous action which got the error will NOT be re-attempted.

In an endless loop usb tryreset() tries to reset the given usb device via the macro resetusb().

This macro calls <u>resetusb()</u> with debugging information.

_resetusb() sends a command via the operating system dependent function os_resetusb() and - if successful - reinitializes the device. os_resetusb() returns -2 to indicate a broken device and all structures should be removed for it.

In that case, the loop is terminated, an error message is produced and usb_tryreset() returns -1.

In case resetusb() has success, the endless loop is left via a return 0 (success).

If the return value from resetusb() is -1, the loop is continued with the next try.

If the global variable **reset_stop** is set directly when the function is called or after each try, **usb_tryreset()** stops working and returns -1.

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

Definition at line 465 of file usb.c.

References ckb_err, ckb_info, reset_stop, and resetusb.

Referenced by _setupusb(), and cmd_fwupdate().

```
465
466
        if (reset stop)
467
            return -1;
        ckb_info("Attempting reset...\n");
469
        while(1){
            int res = resetusb(kb);
470
471
            if(!res){
                ckb_info("Reset success\n");
472
473
                return 0;
474
475
            if(res == -2 || reset_stop)
476
                break;
477
478
        ckb_err("Reset failed. Disconnecting.\n");
479
        return -1;
480 }
```

Here is the caller graph for this function:



5.37.1.12 const char* vendor_str (short vendor)

uncomment the following Define to see USB packets sent to the device

vendor_str returns "corsair" iff the given vendor argument is equal to V_CORSAIR (0x1bc) else it returns ""

Attention

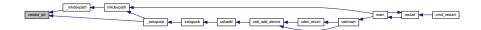
There is also a string defined V_CORSAIR_STR, which returns the device number as string in hex "1b1c".

Definition at line 43 of file usb.c.

References V_CORSAIR.

Referenced by _mkdevpath(), and _setupusb().

Here is the caller graph for this function:



5.37.2 Variable Documentation

5.37.2.1 int features_mask = -1

features_mask Mask of features to exclude from all devices

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

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

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

Definition at line 35 of file usb.c.

Referenced by _setupusb(), and main().

5.37.2.2 int hwload_mode

Definition at line 7 of file device.c.

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

5.37.2.3 volatile int reset_stop = 0

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

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

Definition at line 25 of file usb.c.

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

5.37.2.4 pthread_mutex_t usbmutex = PTHREAD_MUTEX_INITIALIZER

usbmutex is a never referenced mutex!

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

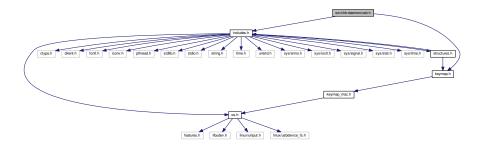
Definition at line 17 of file usb.c.

5.38 src/ckb-daemon/usb.h File Reference

Definitions for using USB interface.

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

Include dependency graph for usb.h:



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



Macros

• #define V_CORSAIR 0x1b1c

For the following Defines please see "Detailed Description".

- #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 || (kb)->p
- #define P_K70 0x1b13
- #define P K70 STR "1b13"
- #define P_K70_NRGB 0x1b09
- #define P_K70_NRGB_STR "1b09"
- #define P K70 LUX 0x1b33
- #define P_K70_LUX_STR "1b33"
- #define P K70 LUX NRGB 0x1b36
- #define P K70 LUX NRGB STR "1b36"
- #define P K70 RFIRE 0x1b38
- #define P_K70_RFIRE_STR "1b38"
- #define P_K70_RFIRE_NRGB 0x1b3a
- #define P K70 RFIRE NRGB STR "1b3a"
- #define IS_K70(kb) ((kb)->vendor == V_CORSAIR && ((kb)->product == P_K70 || (kb)->product == P_K70_NRGB || (kb)->product == P_K70_RFIRE || (kb)->product == P_K70_RFIRE_NRGB || (kb)->product == P_K70_LUX || (kb)->product == P_K70_LUX_NRGB))
- #define P_K95 0x1b11
- #define P K95 STR "1b11"
- #define P K95 NRGB 0x1b08
- #define P K95 NRGB STR "1b08"
- #define P K95 PLATINUM 0x1b2d
- #define P_K95_PLATINUM_STR "1b2d"
- #define IS_K95(kb) ((kb)->vendor == V_CORSAIR && ((kb)->product == P_K95 || (kb)->product == P_K95-NRGB || (kb)->product == P_K95_PLATINUM))
- #define P STRAFE 0x1b20
- #define P STRAFE STR "1b20"
- #define P_STRAFE_NRGB 0x1b15
- #define P_STRAFE_NRGB_STR "1b15"
- #define IS_STRAFE(kb) ((kb)->vendor == V_CORSAIR && ((kb)->product == P_STRAFE || (kb)->product == P_STRAFE NRGB))
- #define P_M65 0x1b12
- #define P M65 STR "1b12"
- #define P_M65_PRO 0x1b2e
- #define P_M65_PRO_STR "1b2e"
- #define IS_M65(kb) ((kb)->vendor == V_CORSAIR && ((kb)->product == P_M65 || (kb)->product == P_M65 || (kb)->p
- #define P SABRE O 0x1b14 /* optical */
- #define P_SABRE_O_STR "1b14"
- #define P_SABRE_L 0x1b19 /* laser */
- #define P_SABRE_L_STR "1b19"
- #define P_SABRE_N 0x1b2f /* new? */
- #define P_SABRE_N_STR "1b2f"
- #define P_SABRE_O2 0x1b32 /* Observed on a CH-9000111-EU model SABRE */
- #define P_SABRE_O2_STR "1b32"
- #define P HARPOON 0x1b3c /* Harpoon test */
- #define P_HARPOON_STR "1b3c"
- #define IS_SABRE(kb) ((kb)->vendor == V_CORSAIR && ((kb)->product == P_SABRE_O || (kb)->product == P_SABRE_L || (kb)->product == P_SABRE_N || (kb)->product == P_HARPOON))
- #define P SCIMITAR 0x1b1e
- #define P_SCIMITAR_STR "1b1e"

- #define P_SCIMITAR_PRO 0x1b3e
- #define P_SCIMITAR_PRO_STR "1b3e"
- #define IS_SCIMITAR(kb) ((kb)->vendor == V_CORSAIR && ((kb)->product == P_SCIMITAR || (kb)->product == P_SCIMITAR PRO))
- #define IS_RGB(vendor, product) ((vendor) == (V_CORSAIR) && (product) != (P_K65_NRGB) && (product) != (P_K70_NRGB) && (product) != (P_K95_NRGB))

RGB vs non-RGB test (note: non-RGB Strafe is still considered "RGB" in that it shares the same protocol. The difference is denoted with the "monochrome" feature).

#define IS_MONOCHROME(vendor, product) ((vendor) == (V_CORSAIR) && (product) == (P_STRAFE_N-RGB))

The difference between non RGB and monochrome is, that monochrome has lights, but just in one color. nonRGB has no lights. Change this if new **monochrome** devices are added.

#define IS RGB DEV(kb) IS RGB((kb)->vendor, (kb)->product)

For calling with a usbdevice*, vendor and product are extracted and IS_RGB() is returned.

• #define IS_MONOCHROME_DEV(kb) IS_MONOCHROME((kb)->vendor, (kb)->product)

For calling with a usbdevice*, vendor and product are extracted and IS_MONOCHROME() is returned.

#define IS_FULLRANGE(kb) (IS_RGB((kb)->vendor, (kb)->product) && (kb)->product != P_K65 && (kb)->product != P_K70 && (kb)->product != P_K95)

Full color range (16.8M) vs partial color range (512)

#define IS_MOUSE(vendor, product) ((vendor) == (V_CORSAIR) && ((product) == (P_M65) || (product) == (P_M65_PRO) || (product) == (P_SABRE_O) || (product) == (P_SABRE_L) || (product) == (P_SCIMITAR) || (product) == (P_SCIMITAR_PRO) || (product) == (P_SABRE_O2)))

Mouse vs keyboard test.

#define IS_MOUSE_DEV(kb) IS_MOUSE((kb)->vendor, (kb)->product)

For calling with a usbdevice*, vendor and product are extracted and IS MOUSE() is returned.

#define DELAY_SHORT(kb) usleep((int)(kb)->usbdelay * 1000)

USB delays for when the keyboards get picky about timing That was the original comment, but it is used anytime. The short delay is used before any send or receive.

• #define DELAY MEDIUM(kb) usleep((int)(kb)->usbdelay * 10000)

the medium delay is used after sending a command before waiting for the answer.

#define DELAY_LONG(kb) usleep(100000)

The longest delay takes place where something went wrong (eg when resetting the device)

#define USB_DELAY_DEFAULT 5

This constant is used to initialize **kb->usbdelay**. It is used in many places (see macros above) but often also overwritten to the fixed value of 10. Pure Hacker code.

#define resetusb(kb) _resetusb(kb, __FILE_NOPATH__, __LINE__)

resetusb() is just a macro to call _resetusb() with debuggin constants (file, lineno)

- #define usbsend(kb, messages, count) _usbsend(kb, messages, count, __FILE_NOPATH__, __LINE__)
 usbsend macro is used to wrap _usbsend() with debugging information (file and lineno)
- #define usbrecv(kb, out_msg, in_msg) _usbrecv(kb, out_msg, in_msg, __FILE_NOPATH__, __LINE__)
 usbrecv macro is used to wrap _usbrecv() with debugging information (file and lineno)
- #define nk95cmd(kb, command) _nk95cmd(kb, (command) >> 16 & 0xFF, (command) & 0xFFFF, __FILE_-NOPATH , LINE)

nk95cmd() macro is used to wrap _nk95cmd() with debugging information (file and lineno). the command structure is different:

Just the bits 23..16 are used as bits 7..0 for bRequest

Bits 15..0 are used as wValue

#define NK95_HWOFF 0x020030

Hardware-specific commands for the K95 nonRGB,.

#define NK95_HWON 0x020001

Hardware playback on.

• #define NK95_M1 0x140001

Switch to mode 1.

#define NK95_M2 0x140002

Switch to mode 2.

• #define NK95 M3 0x140003

Switch to mode 3.

Functions

const char * vendor str (short vendor)

uncomment the following Define to see USB packets sent to the device

const char * product str (short product)

product_str returns a condensed view on what type of device we have.

• int usbmain ()

Start the USB main loop. Returns program exit code when finished.

• void usbkill ()

Stop the USB system.

void setupusb (usbdevice *kb)

setupusb starts a thread with kb as parameter and _setupusb() as entrypoint.

int os_setupusb (usbdevice *kb)

os_setupusb OS-specific setup for a specific usb device.

void * os inputmain (void *context)

os_inputmain is run in a separate thread and will be detached from the main thread, so it needs to clean up its own resources.

int revertusb (usbdevice *kb)

revertusb sets a given device to inactive (hardware controlled) mode if not a fw-ugrade is indicated

int closeusb (usbdevice *kb)

closeusb Close a USB device and remove device entry.

void os_closeusb (usbdevice *kb)

os_closeusb unclaim it, destroy the udev device and clear data structures at kb

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

_resetusb Reset a USB device.

int os_resetusb (usbdevice *kb, const char *file, int line)

os_resetusb is the os specific implementation for resetting usb

• int usbsend (usbdevice *kb, const uchar *messages, int count, const char *file, int line)

usbsend send a logical message completely to the given device

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

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

• int os_usbsend (usbdevice *kb, const uchar *out_msg, int is_recv, const char *file, int line)

os_usbsend sends a data packet (MSG_SIZE = 64) Bytes long

• int os_usbrecv (usbdevice *kb, uchar *in_msg, const char *file, int line)

os_usbrecv receives a max MSGSIZE long buffer from usb device

void os_sendindicators (usbdevice *kb)

os_sendindicators update the indicators for the special keys (Numlock, Capslock and what else?)

• int _nk95cmd (usbdevice *kb, uchar bRequest, ushort wValue, const char *file, int line)

_nk95cmd If we control a non RGB keyboard, set the keyboard via ioctl with usbdevfs_ctrltransfer

int usb_tryreset (usbdevice *kb)

usb_tryreset does what the name means: Try to reset the usb via resetusb()

5.38.1 Detailed Description

Vendor/product codes

The list of defines in the first part of the file describes the various types of equipment from Corsair and summarizes them according to specific characteristics.

Each device type is described with two defines:

- · On the one hand the device ID with which the device can be recognized on the USB as a short
- and on the other hand the same representation as a string, but without leading "0x".

First entry-pair is the Provider ID (vendorID) from Corsair.

Block No.	contains	Devices are bundled via
1	The first block contains the	In summary, they can be queried
	K65-like keyboards, regardless of	using the macro IS_K65().
	their properties (RGB,).	
2	the K70-like Keyboards with all	summarized by IS_K70().
	their configuration types	
3	the K95 series keyboards	collected with the macro IS_K95().
4	strafe keyboards	IS_STRAFE()
5	M65 mice with and without RGB	IS_M65()
6	The SABRE and HARPOON mice.	IS_SABRE()
	Maybe this will be divided int two	
	different blocks later because of	
	different nummber of special keys	
7	The Scimitar mouse devices	IS_SCIMITAR()

Definition in file usb.h.

5.38.2 Macro Definition Documentation

5.38.2.1 #define DELAY_LONG(kb) usleep(100000)

Definition at line 156 of file usb.h.

Referenced by _resetusb(), _usbrecv(), _usbsend(), cmd_hwload_kb(), cmd_hwload_mouse(), cmd_hwsave_kb(), and cmd_hwsave_mouse().

5.38.2.2 #define DELAY_MEDIUM(kb) usleep((int)(kb)->usbdelay * 10000)

Definition at line 153 of file usb.h.

Referenced by _usbrecv(), and setactive_kb().

5.38.2.3 #define DELAY_SHORT(kb) usleep((int)(kb)->usbdelay * 1000)

Definition at line 150 of file usb.h.

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

5.38.2.4 #define IS_FULLRANGE(kb) (IS_RGB((kb)->vendor, (kb)->product) && (kb)->product != P_K65 && (kb)->product != P_K70 && (kb)->product != P_K95)

Definition at line 139 of file usb.h.

Referenced by readcmd(), and updatergb_kb().

5.38.2.5 #define IS_K65(*kb*) ((kb)->vendor == V_CORSAIR && ((kb)->product == P_K65 || (kb)->product == P_K65_NRGB || (kb)->product == P_K65_LUX || (kb)->product == P_K65_RFIRE))

Definition at line 49 of file usb.h.

Referenced by has key().

5.38.2.6 #define IS_K70(kb) ((kb)->vendor == V_CORSAIR && ((kb)->product == P_K70 || (kb)->product == P_K70_NRGB || (kb)->product == P_K70_RFIRE || (kb)->product == P_K70_LUX || (kb)->product == P_K70_LUX_NRGB))

Definition at line 63 of file usb.h.

5.38.2.7 #define IS_K95(kb) ((kb)->vendor == V_CORSAIR && ((kb)->product == P_K95 || (kb)->product == P_K95_NRGB || (kb)->product == P_K95_PLATINUM))

Definition at line 71 of file usb.h.

Referenced by cmd hwload kb(), cmd hwsave kb(), and has key().

5.38.2.8 #define IS_M65(*kb*) ((kb)->vendor == V_CORSAIR && ((kb)->product == P_M65 || (kb)->product == P_M65 PRO))

Definition at line 83 of file usb.h.

Referenced by isblack().

5.38.2.9 #define IS_MONOCHROME(vendor, product) ((vendor) == (V CORSAIR) && (product) == (P STRAFE NRGB))

Definition at line 130 of file usb.h.

Referenced by _setupusb().

5.38.2.10 #define IS_MONOCHROME_DEV(kb) IS MONOCHROME((kb)->vendor, (kb)->product)

Definition at line 136 of file usb.h.

5.38.2.11 #define IS_MOUSE(vendor, product) ((vendor) == (V_CORSAIR) && ((product) == (P_M65) || (product) == (P_M65_PRO) || (product) == (P_SABRE_O) || (product) == (P_SABRE_L) || (product) == (P_SABRE_N) || (product) == (P_SCIMITAR) || (product) == (P_SCIMITAR_PRO) || (product) == (P_SABRE_O2)))

Definition at line 142 of file usb.h.

Referenced by setupusb(), get vtable(), has key(), and os inputmain().

5.38.2.12 #define IS_MOUSE_DEV(kb) IS MOUSE((kb)->vendor, (kb)->product)

Definition at line 145 of file usb.h.

Referenced by readcmd().

5.38.2.13 #define IS_RGB(vendor, product) ((vendor) == (V_CORSAIR) && (product) != (P_K65_NRGB) && (product) != (P_K70_NRGB) && (product) != (P_K95_NRGB))

Definition at line 125 of file usb.h.

Referenced by _setupusb(), get_vtable(), and os_inputmain().

5.38.2.14 #define IS_RGB_DEV(kb) IS_RGB((kb)->vendor, (kb)->product)

Definition at line 133 of file usb.h.

5.38.2.15 #define IS_SABRE(kb) ((kb)->vendor == V_CORSAIR && ((kb)->product == P_SABRE_O || (kb)->product == P_SABRE_O || (k

Definition at line 95 of file usb.h.

Referenced by has_key(), loadrgb_mouse(), and savergb_mouse().

5.38.2.16 #define IS_SCIMITAR(kb) ((kb)->vendor == V_CORSAIR && ((kb)->product == P_SCIMITAR || (kb)->product == P_SCIMITAR_PRO))

Definition at line 101 of file usb.h.

Referenced by has_key(), loadrgb_mouse(), and savergb_mouse().

5.38.2.17 #define IS_STRAFE(kb) ((kb)->vendor == V_CORSAIR && ((kb)->product == P_STRAFE || (kb)->product == P_STRAFE || (kb)->pro

Definition at line 77 of file usb.h.

Referenced by savergb_kb().

5.38.2.18 #define NK95_HWOFF 0x020030

See Also

usb2.0 documentation for details. Set Hardware playback off

Definition at line 302 of file usb.h.

Referenced by start_kb_nrgb().

5.38.2.19 #define NK95_HWON 0x020001

Definition at line 305 of file usb.h.

Referenced by revertusb().

5.38.2.20 #define NK95_M1 0x140001

Definition at line 308 of file usb.h.

Referenced by setmodeindex nrgb().

5.38.2.21 #define NK95_M2 0x140002

Definition at line 311 of file usb.h.

Referenced by setmodeindex_nrgb().

```
5.38.2.22 #define NK95_M3 0x140003
Definition at line 314 of file usb.h.
Referenced by setmodeindex nrgb().
5.38.2.23 #define nk95cmd( kb, command ) _nk95cmd(kb, (command) >> 16 & 0xFF, (command) & 0xFFFF,
          __FILE_NOPATH__, __LINE__)
Definition at line 297 of file usb.h.
Referenced by revertusb(), setmodeindex_nrgb(), and start_kb_nrgb().
5.38.2.24 #define P_HARPOON 0x1b3c /* Harpoon test */
Definition at line 93 of file usb.h.
Referenced by product_str().
5.38.2.25 #define P_HARPOON_STR "1b3c"
Definition at line 94 of file usb.h.
5.38.2.26 #define P_K65 0x1b17
Definition at line 41 of file usb.h.
Referenced by product_str().
5.38.2.27 #define P_K65_LUX 0x1b37
Definition at line 45 of file usb.h.
Referenced by product_str().
5.38.2.28 #define P_K65_LUX_STR "1b37"
Definition at line 46 of file usb.h.
5.38.2.29 #define P_K65_NRGB 0x1b07
Definition at line 43 of file usb.h.
Referenced by product_str().
5.38.2.30 #define P_K65_NRGB_STR "1b07"
Definition at line 44 of file usb.h.
5.38.2.31 #define P_K65_RFIRE 0x1b39
Definition at line 47 of file usb.h.
```

Referenced by product_str().

5.38.2.32 #define P_K65_RFIRE_STR "1b39"

Definition at line 48 of file usb.h.

5.38.2.33 #define P_K65_STR "1b17"

Definition at line 42 of file usb.h.

5.38.2.34 #define P_K70 0x1b13

Definition at line 51 of file usb.h.

Referenced by product_str().

5.38.2.35 #define P_K70_LUX 0x1b33

Definition at line 55 of file usb.h.

Referenced by product_str().

5.38.2.36 #define P_K70_LUX_NRGB 0x1b36

Definition at line 57 of file usb.h.

Referenced by product_str().

5.38.2.37 #define P_K70_LUX_NRGB_STR "1b36"

Definition at line 58 of file usb.h.

5.38.2.38 #define P_K70_LUX_STR "1b33"

Definition at line 56 of file usb.h.

5.38.2.39 #define P_K70_NRGB 0x1b09

Definition at line 53 of file usb.h.

Referenced by product_str().

5.38.2.40 #define P_K70_NRGB_STR "1b09"

Definition at line 54 of file usb.h.

5.38.2.41 #define P_K70_RFIRE 0x1b38

Definition at line 59 of file usb.h.

Referenced by product_str().

5.38.2.42 #define P_K70_RFIRE_NRGB 0x1b3a

Definition at line 61 of file usb.h.

Referenced by product_str().

5.38.2.43 #define P_K70_RFIRE_NRGB_STR "1b3a"

Definition at line 62 of file usb.h.

5.38.2.44 #define P_K70_RFIRE_STR "1b38"

Definition at line 60 of file usb.h.

5.38.2.45 #define P_K70_STR "1b13"

Definition at line 52 of file usb.h.

5.38.2.46 #define P_K95 0x1b11

Definition at line 65 of file usb.h.

Referenced by product_str().

5.38.2.47 #define P_K95_NRGB 0x1b08

Definition at line 67 of file usb.h.

Referenced by _nk95cmd(), and product_str().

5.38.2.48 #define P_K95_NRGB_STR "1b08"

Definition at line 68 of file usb.h.

5.38.2.49 #define P_K95_PLATINUM 0x1b2d

Definition at line 69 of file usb.h.

Referenced by product_str().

5.38.2.50 #define P_K95_PLATINUM_STR "1b2d"

Definition at line 70 of file usb.h.

5.38.2.51 #define P_K95_STR "1b11"

Definition at line 66 of file usb.h.

5.38.2.52 #define P_M65 0x1b12

Definition at line 79 of file usb.h.

Referenced by product_str().

```
5.38.2.53 #define P_M65_PRO 0x1b2e
Definition at line 81 of file usb.h.
Referenced by product_str().
5.38.2.54 #define P_M65_PRO_STR "1b2e"
Definition at line 82 of file usb.h.
5.38.2.55 #define P_M65_STR "1b12"
Definition at line 80 of file usb.h.
5.38.2.56 #define P_SABRE_L 0x1b19 /* laser */
Definition at line 87 of file usb.h.
Referenced by product_str().
5.38.2.57 #define P_SABRE_L_STR "1b19"
Definition at line 88 of file usb.h.
5.38.2.58 #define P_SABRE_N 0x1b2f /* new? */
Definition at line 89 of file usb.h.
Referenced by product_str().
5.38.2.59 #define P SABRE N STR "1b2f"
Definition at line 90 of file usb.h.
5.38.2.60 #define P_SABRE_O 0x1b14 /* optical */
Definition at line 85 of file usb.h.
Referenced by product_str().
5.38.2.61 #define P_SABRE_02 0x1b32 /* Observed on a CH-9000111-EU model SABRE */
Definition at line 91 of file usb.h.
Referenced by product_str().
5.38.2.62 #define P SABRE O2 STR "1b32"
Definition at line 92 of file usb.h.
5.38.2.63 #define P_SABRE_O_STR "1b14"
Definition at line 86 of file usb.h.
```

```
5.38.2.64 #define P_SCIMITAR 0x1b1e
Definition at line 97 of file usb.h.
Referenced by product_str().
5.38.2.65 #define P_SCIMITAR_PRO 0x1b3e
Definition at line 99 of file usb.h.
Referenced by product_str().
5.38.2.66 #define P_SCIMITAR_PRO_STR "1b3e"
Definition at line 100 of file usb.h.
5.38.2.67 #define P_SCIMITAR_STR "1b1e"
Definition at line 98 of file usb.h.
5.38.2.68 #define P_STRAFE 0x1b20
Definition at line 73 of file usb.h.
Referenced by product_str().
5.38.2.69 #define P_STRAFE_NRGB 0x1b15
Definition at line 75 of file usb.h.
Referenced by product_str().
5.38.2.70 #define P_STRAFE_NRGB_STR "1b15"
Definition at line 76 of file usb.h.
5.38.2.71 #define P_STRAFE_STR "1b20"
Definition at line 74 of file usb.h.
5.38.2.72 #define resetusb( kb ) resetusb(kb, FILE NOPATH , _LINE_)
Definition at line 215 of file usb.h.
Referenced by usb_tryreset().
5.38.2.73 #define USB_DELAY_DEFAULT 5
Definition at line 161 of file usb.h.
Referenced by _setupusb(), and start_dev().
5.38.2.74 #define usbrecv( kb, out_msg, in_msg ) _usbrecv(kb, out_msg, in_msg, __FILE_NOPATH__, _LINE__)
```

Parameters

kb	THE usbdevice*
IN]	out_msg What information does the caller want from the device?
OUT]	in_msg Here comes the answer; The names represent the usb view, not the view of this
	function! So INput from usb is OUTput of this function.

Definition at line 257 of file usb.h.

Referenced by cmd_hwload_kb(), cmd_hwload_mouse(), getfwversion(), hwloadmode(), loaddpi(), loaddpi(), and loadrgb_mouse().

5.38.2.75 #define usbsend(kb, messages, count) usbsend(kb, messages, count, FILE NOPATH , LINE_)

Parameters

kb	THE usbdevice*
IN]	messages a Pointer to the first byte of the logical message
IN]	count how many MSG_SIZE buffers is the logical message long?

Definition at line 240 of file usb.h.

Referenced by cmd_hwsave_kb(), cmd_hwsave_mouse(), cmd_pollrate(), fwupdate(), loadrgb_kb(), savedpi(), savergb_kb(), savergb_mouse(), setactive_mouse(), updatedpi(), updatergb_kb(), and updatergb_mouse().

5.38.2.76 #define V CORSAIR 0x1b1c

Warning

When adding new devices please update src/ckb/fwupgradedialog.cpp as well.

It should contain the same vendor/product IDs for any devices supporting firmware updates.

In the same way, all other corresponding files have to be supplemented or modified: Currently known for this are **usb linux.c** and **usb mac.c**

Definition at line 38 of file usb.h.

Referenced by usb_add_device(), and vendor_str().

5.38.2.77 #define V_CORSAIR_STR "1b1c"

Definition at line 39 of file usb.h.

Referenced by udev_enum(), and usb_add_device().

5.38.3 Function Documentation

5.38.3.1 int_nk95cmd (usbdevice * kb, uchar bRequest, ushort wValue, const char * file, int line)

Parameters

kb	THE usbdevice*	
bRequest	he byte array with the usb request	
wValue	a usb wValue	

file	for error message	
line	for error message	

Returns

1 (true) on failure, 0 (false) on success.

To send control packets to a non RGB non color K95 Keyboard, use this function. Normally it is called via the nk95cmd() macro.

If it is the wrong device for which the function is called, 0 is returned and nothing done. Otherwise a usbdevfs_ctrltransfer structure is filled and an USBDEVFS_CONTROL ioctl() called.

bRequest- Type	bRequest	wValue	EP	size	Timeout	data
0x40	see table below to switch hardware- modus at Keyboard	wValue	device	MSG_SIZE	5ms	the message buffer pointer
Host to Device, Type=Vendor, Recipi- ent=Device	bRequest parameter	given wValue Parameter	device 0	0 data to write	5000	null

If a 0 or a negative error number is returned by the ioctl, an error message is shown depending on the error or "No data written" if retval was 0. In either case 1 is returned to indicate the error. If the ioctl returned a value > 0, 0 is returned to indicate no error.

Currently the following combinations for bRequest and wValue are used:

Device	what it might to	constant	bRequest	wValue
	do			
non RGB Keyboard	set HW-modus on	HWON	0x0002	0x0030
	(leave the ckb			
	driver)			
non RGB Keyboard	set HW-modus off	HWOFF	0x0002	0x0001
	(initialize the ckb			
	driver)			
non RGB Keyboard	set light modus M1	NK95_M1	0x0014	0x0001
	in single-color			
	keyboards			
non RGB Keyboard	set light modus M2	NK95_M2	0x0014	0x0002
	in single-color			
	keyboards			
non RGB Keyboard	set light modus M3	NK95_M3	0x0014	0x0003
	in single-color			
	keyboards			

See Also

usb.h

Definition at line 187 of file usb_linux.c.

References ckb_err_fn, usbdevice::handle, P_K95_NRGB, and usbdevice::product.

5.38.3.2 int_resetusb (usbdevice *kb, const char *file, int line)

Parameters

kb	THE usbdevice*	
file	ename for error messages	
line	line where it is called for error messages	

Returns

Returns 0 on success, -1 if device should be removed

_resetusb Reset a USB device.

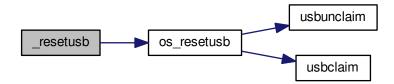
First reset the device via os_resetusb() after a long delay (it may send something to the host). If this worked (retval == 0), give the device another long delay Then perform the initialization via the device specific start() function entry in kb->vtable and if this is successful also, return the result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

Definition at line 426 of file usb.c.

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

```
426
427
        // Perform a USB reset
428
        DELAY_LONG(kb);
429
        int res = os_resetusb(kb, file, line);
430
        if(res)
        return res;
DELAY_LONG(kb);
431
432
433
        // Re-initialize the device
        if (kb->vtable->start(kb, kb->active) != 0)
434
435
              eturn -1;
436
        if(kb->vtable->updatergb(kb, 1) != 0)
437
            return -1;
        return 0;
438
439 }
```

Here is the call graph for this function:



5.38.3.3 int_usbrecv (usbdevice * kb, const uchar * out_msg, uchar * in_msg, const char * file, int line)

Parameters

	kb	THE usbdevice*
	IN]	out_msg What information does the caller want from the device?
	OUT]	in_msg Here comes the answer; The names represent the usb view, not the
		view of this function! So INput from usb is OUTput of this function.
	IN]	file for debugging
	IN]	line for debugging
in	reset_stop	global variable is read

Returns

number of bytes read or zero on failure.

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

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

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

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

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

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

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

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

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

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

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

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

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

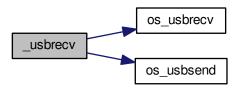
Definition at line 599 of file usb.c.

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

```
599
600
         // Try a maximum of 3 times
601
         for(int try = 0; try < 5; try++) {</pre>
             // Send the output message
DELAY_SHORT(kb);
602
603
604
             int res = os_usbsend(kb, out_msg, 1, file, line);
             if(res == 0)
605
                 return 0;
606
607
             else if(res == -1){
608
                  // Retry on temporary failure
609
                  if(reset_stop)
610
                      return 0;
                  DELAY_LONG(kb);
611
```

```
continue;
             // Wait for the response
614
            DELAY_MEDIUM(kb);
615
            res = os_usbrecv(kb, in_msg, file, line);
if(res == 0)
616
617
                return 0;
618
619
             else if(res != -1)
620
                 return res;
621
            if(reset_stop || hwload_mode != 2)
622
                 return 0:
            DELAY_LONG(kb);
623
624
625
626
        ckb_err_fn("Too many send/recv failures. Dropping.\n", file, line);
627
        return 0;
628 1
```

Here is the call graph for this function:



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

Parameters

	kb	THE usbdevice*
	IN]	messages a Pointer to the first byte of the logical message
	IN]	count how many MSG_SIZE buffers is the logical message long?
	IN]	file for debugging
	IN]	line for debugging
in	reset_stop	global variable is read

Returns

number of Bytes sent (ideal == count * MSG_SIZE); 0 if a block could not be sent and it was not a timeout OR **reset_stop** was required or **hwload_mode** is not set to "always"

_usbsend send a logical message completely to the given device

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

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

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

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

Attention

This means that the buffer given as argument must be n * MSG SIZE Byte long.

An essential constant parameter which is relevant for os_usbsend() only is is_recv = 0, which means sending. Now it gets a little complicated again:

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

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

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

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

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

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

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

Definition at line 532 of file usb.c.

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

```
532
533
        int total sent = 0:
        for(int i = 0; i < count; i++) {</pre>
534
535
             // Send each message via the OS function
536
             while (1) {
537
                 DELAY_SHORT (kb);
                 int res = os_usbsend(kb, messages + i * MSG_SIZE, 0, file, line);
538
                 <u>if</u>(res == 0)
539
540
                     return 0;
                 else if (res != -1) {
541
542
                     total_sent += res;
543
544
                 // Stop immediately if the program is shutting down or hardware load is set to tryonce
545
                 if(reset_stop || hwload_mode != 2)
                     return 0;
547
                 // Retry as long as the result is temporary failure
548
549
                 DELAY_LONG(kb);
550
551
552
        return total_sent;
553 }
```

Here is the call graph for this function:



5.38.3.5 int closeusb (usbdevice * kb)

Parameters

IN,OUT]	kb
---------	----

Returns

Returns 0 (everytime. No error handling is done!)

closeusb Close a USB device and remove device entry.

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

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

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

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

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

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

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

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

See Also

the hack note with rmdevpath())

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

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

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

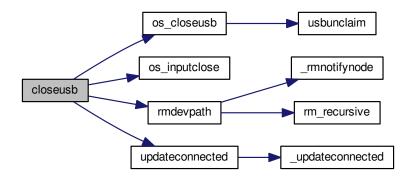
Definition at line 673 of file usb.c.

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

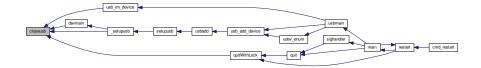
Referenced by _setupusb(), devmain(), quitWithLock(), and usb_rm_device().

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

Here is the call graph for this function:



Here is the caller graph for this function:



5.38.3.6 void os_closeusb (usbdevice * kb)

Parameters

```
IN,OUT] kb THE usbdevice*
```

os_closeusb unclaim it, destroy the udev device and clear data structures at kb

os closeusb is the linux specific implementation for closing an active usb port.

If a valid handle is given in the kb structure, the usb port is unclaimed (usbunclaim()).

The device in unrefenced via library function udev_device_unref().

handle, udev and the first char of kbsyspath are cleared to 0 (empty string for kbsyspath).

Definition at line 422 of file usb_linux.c.

References usbdevice::handle, INDEX_OF, kbsyspath, keyboard, usbdevice::udev, and usbunclaim().

Referenced by closeusb().

```
422
423
         if(kb->handle){
424
             usbunclaim(kb, 0);
close(kb->handle - 1);
425
426
427
        if (kb->udev)
428
             udev_device_unref(kb->udev);
429
        kb->handle = 0;
430
        kb->udev = 0;
431
        kbsyspath[INDEX_OF(kb, keyboard)][0] = 0;
432 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.38.3.7 void* os_inputmain (void * context)

Parameters

context	THE usbdevice*; Because os_inputmain() is started as a new thread, its formal parameter
	is named "context".

Returns

null

os_inputmain is run in a separate thread and will be detached from the main thread, so it needs to clean up its own resources.

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

- 1. set up an URB (Userspace Ressource Buffer) to communicate with the USBDEVFS * ioctl()s
- 2. perform the ioctl()
- 3. interpretate the information got into the URB buffer or handle error situations and retry operation or leave the endless loop
- 4. inform the os about the data
- 5. loop endless via 2.
- 6. if endless loop has gone, deinitalize the interface, free buffers etc.
- 7. return null

Here the actions in detail:

Monitor input transfers on all endpoints for non-RGB devices For RGB, monitor all but the last, as it's used for input/output

Get an usbdevfs_urb data structure and clear it via memset()

Hopefully the buffer lengths are equal for all devices with congruent types. You can find out the correctness for your device with Isusb –v or similar on macOS. Currently the following combinations are known and implemented:

device detect with macro combination		endpoint #	buffer-length	
each	none	0	8	
RGB Mouse	IS_RGB && IS_MOUSE	1	10	
RGB Keyboard	IS_RGB && !IS_MOUSE	1	21	
RGB Mouse or Keyboard IS_RGB		2	MSG_SIZE (64)	
non RGB Mouse or	!IS_RGB	1	4	
Keyboard				
non RGB Mouse or !IS_RGB		2	15	
Keyboard				

Now submit all the URBs via ioctl(USBDEVFS_SUBMITURB) with type USBDEVFS_URB_TYPE_INTERRUPT (the endpoints are defined as type interrupt). Endpoint number is 0x80..0x82 or 0x83, depending on the model.

The userSpaceFS knows the URBs now, so start monitoring input

if the ioctl returns something != 0, let's have a deeper look what happened. Broken devices or shutting down the entire system leads to closing the device and finishing this thread.

If just an EPIPE ocurred, give the device a CLEAR_HALT and resubmit the URB.

A correct REAPURB returns a Pointer to the URB which we now have a closer look into. Lock all following actions with imutex.

Process the input depending on type of device. Interprete the actual size of the URB buffer

device	detect with macro	seems to be	actual	function called
	combination	endpoint #	buffer-length	

mouse (RGB and	IS_MOUSE	nA	8, 10 or 11	hid_mouse
non RGB)				translate()
mouse (RGB and	IS_MOUSE	nA	MSG_SIZE (64)	corsair
non RGB)				mousecopy()
RGB Keyboard	IS_RGB &&	1	8 (BIOS Mode)	hid_kb_translate()
	!IS_MOUSE			
RGB Keyboard	IS_RGB &&	2	5 or 21, KB	hid_kb_translate()
	!IS_MOUSE		inactive!	
RGB Keyboard	IS_RGB &&	3?	MSG_SIZE	corsair_kbcopy()
	!IS_MOUSE			
non RGB Keyboard	!IS_RGB &&	nA	nA	hid_kb_translate()
	!IS_MOUSE			

The input data is transformed and copied to the kb structure. Now give it to the OS and unlock the imutex afterwards.

Re-submit the URB for the next run.

If the endless loop is terminated, clean up by discarding the URBs via ioctl(USBDEVFS_DISCARDURB), free the URB buffers and return a null pointer as thread exit code.

Definition at line 232 of file usb_linux.c.

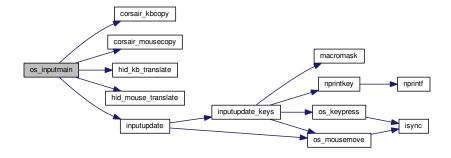
References usbdevice::active, ckb_info, corsair_kbcopy(), corsair_mousecopy(), devpath, usbdevice::epcount, usbdevice::handle, hid_kb_translate(), hid_mouse_translate(), imutex, INDEX_OF, usbdevice::input, inputupdate(), IS_MOUSE, IS_RGB, keyboard, usbinput::keys, MSG_SIZE, usbdevice::product, usbinput::rel_x, usbinput::rel_y, and usbdevice::vendor.

Referenced by _setupusb().

```
232
233
        usbdevice* kb = context;
        int fd = kb->handle - 1:
234
235
        short vendor = kb->vendor, product = kb->product;
        int index = INDEX_OF(kb, keyboard);
        ckb_info("Starting input thread for %s%d\n", devpath, index);
238
243
       int urbcount = IS_RGB(vendor, product) ? (kb->epcount - 1) : kb->
      epcount:
244
246
        struct usbdevfs_urb urbs[urbcount];
247
        memset(urbs, 0, sizeof(urbs));
248
2.62
        urbs[0].buffer_length = 8;
263
        if(IS_RGB(vendor, product)){
            if(IS_MOUSE(vendor, product))
    urbs[1].buffer_length = 10;
264
265
                urbs[1].buffer_length = 21;
267
268
            urbs[2].buffer_length = MSG_SIZE;
269
            if (urbcount != 3)
                urbs[urbcount - 1].buffer_length = MSG_SIZE;
270
       } else {
272
           urbs[1].buffer_length = 4;
273
            urbs[2].buffer_length = 15;
274
275
        for (int i = 0; i < urbcount; i++) {
278
279
           urbs[i].type = USBDEVFS_URB_TYPE_INTERRUPT;
            urbs[i].endpoint = 0x80 \mid (i + 1);
281
            urbs[i].buffer = malloc(urbs[i].buffer_length);
282
            ioctl(fd, USBDEVFS_SUBMITURB, urbs + i);
283
       }
284
        while (1) {
286
287
            struct usbdevfs_urb* urb = 0;
288
291
            if (ioctl(fd, USBDEVFS_REAPURB, &urb)) {
                 if (errno == ENODEV || errno == ENOENT || errno == ESHUTDOWN)
292
293
                     // Stop the thread if the handle closes
294
                    break;
                else if(errno == EPIPE && urb){
297
                    ioctl(fd, USBDEVFS_CLEAR_HALT, &urb->endpoint);
298
                    // Re-submit the URB
299
                    if(urb)
                        ioctl(fd, USBDEVFS_SUBMITURB, urb);
300
301
                    urb = 0;
302
                }
```

```
304
308
             if (urb) {
320
                 pthread_mutex_lock(imutex(kb));
321
                 if(IS_MOUSE(vendor, product)){
322
                     switch(urb->actual_length) {
323
                     case 8:
324
                     case 10:
325
                     case 11:
                         // HID mouse input
326
327
                         hid_mouse_translate(kb->input.keys, &kb->
      input.rel_x, &kb->input.rel_y, -(urb->endpoint & 0xF), urb->actual_length, urb->buffer)
328
                         break;
329
                     case MSG_SIZE:
330
                         // Corsair mouse input
331
                         corsair\_mousecopy(kb->input.keys, -(urb->endpoint & 0xF), urb
      ->huffer):
332
333
                 } else if(IS_RGB(vendor, product)){
    switch(urb->actual_length){
334
335
336
                     case 8:
                         // RGB EP 1: 6KRO (BIOS mode) input
337
                         hid_kb_translate(kb->input.keys, -1, urb->actual_length, urb->
338
      buffer);
339
                         break;
340
                     case 21:
341
                     case 5:
                         // RGB EP 2: NKRO (non-BIOS) input. Accept only if keyboard is inactive
342
343
                         if(!kb->active)
                              hid_kb_translate(kb->input.keys, -2, urb->actual_length,
344
      urb->buffer);
345
346
                     case MSG_SIZE:
347
                         // RGB EP 3: Corsair input
                         corsair_kbcopy(kb->input.keys, -(urb->endpoint & 0xF), urb->
348
      buffer);
349
350
                     }
351
                 } else {
352
                     // Non-RGB input
                     hid_kb_translate(kb->input.keys, urb->endpoint & 0xF, urb->
353
      actual length, urb->buffer);
354
357
                 inputupdate(kb);
358
                 pthread_mutex_unlock(imutex(kb));
360
                 ioctl(fd, USBDEVFS_SUBMITURB, urb);
361
                 urb = 0;
             }
362
363
        }
364
368
        ckb_info("Stopping input thread for %s%d\n", devpath, index);
369
        for(int i = 0; i < urbcount; i++) {</pre>
370
             ioctl(fd, USBDEVFS_DISCARDURB, urbs + i);
371
             free(urbs[i].buffer);
372
373
        return 0;
374 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.38.3.8 int os_resetusb (usbdevice * kb, const char * file, int line)

Parameters

kb	THE usbdevice*
file	filename for error messages
line	line where it is called for error messages

Returns

Returns 0 on success, -2 if device should be removed and -1 if reset should by tried again

os_resetusb is the os specific implementation for resetting usb

Try to reset an usb device in a linux user space driver.

- 1. unclaim the device, but do not reconnect the system driver (second param resetting = true)
- 2. reset the device via USBDEVFS_RESET command
- 3. claim the device again. Returns 0 on success, -2 if device should be removed and -1 if reset should by tried again

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

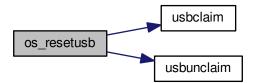
Definition at line 480 of file usb_linux.c.

References usbdevice::handle, TEST_RESET, usbclaim(), and usbunclaim().

Referenced by _resetusb().

```
480 {
481    TEST_RESET(usbunclaim(kb, 1));
482    TEST_RESET(ioctl(kb->handle - 1, USBDEVFS_RESET));
483    TEST_RESET(usbclaim(kb));
484    // Success!
485    return 0;
486 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.38.3.9 void os_sendindicators (usbdevice * kb)

Parameters

kb THE usbdevice*	
---------------------	--

os_sendindicators update the indicators for the special keys (Numlock, Capslock and what else?)

os_sendindicators update the indicators for the special keys (Numlock, Capslock and what else?)

Read the data from kb->ileds ans send them via ioctl() to the keyboard.

bRequest-	bRequest	wValue	EP	size	Timeout	data
Туре						
0x21	0x09	0x0200	Interface 0	MSG_SIZE	timeout	the message
				1 Byte	0,5ms	buffer pointer
Host to	9 = SEND?	specific	0	1	500	struct*
Device,						kb->ileds
Type=Class,						
Recipi-						
ent=Interface						
(why not						
endpoint?)						

The ioctl command is USBDEVFS_CONTROL.

Definition at line 212 of file usb_linux.c.

References ckb_err, usbdevice::handle, and usbdevice::ileds.

Referenced by updateindicators_kb().

Here is the caller graph for this function:



5.38.3.10 int os_setupusb (usbdevice * kb)

Parameters

kb	THE usbdevice*
----	----------------

Returns

0 on success, -1 otherwise.

os_setupusb OS-specific setup for a specific usb device.

Perform the operating system-specific opening of the interface in os_setupusb(). As a result, some parameters should be set in kb (name, serial, fwversion, epcount = number of usb endpoints), and all endpoints should be claimed with usbclaim(). Claiming is the only point where os setupusb() can produce an error (-1).

- · Copy device description and serial
- · Copy firmware version (needed to determine USB protocol)
- · Do some output abaout connecting interfaces
- · Claim the USB interfaces

Todo in these modules a pullrequest is outstanding

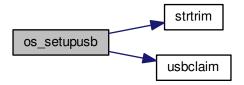
Definition at line 518 of file usb_linux.c.

References ckb_err, ckb_info, ckb_warn, devpath, usbdevice::epcount, FEAT_RGB, usbdevice::fwversion, HAS_-FEATURES, INDEX_OF, KB_NAME_LEN, keyboard, usbdevice::name, usbdevice::serial, SERIAL_LEN, strtrim(), usbdevice::udev, and usbclaim().

Referenced by _setupusb().

```
518
521
        struct udev device* dev = kb->udev;
522
        const char* name = udev_device_get_sysattr_value(dev, "product");
523
524
            strncpy(kb->name, name, KB_NAME_LEN);
525
        strtrim(kb->name);
526
        const char* serial = udev_device_get_sysattr_value(dev, "serial");
527
        if(serial)
528
            strncpy(kb->serial, serial, SERIAL_LEN);
529
        strtrim(kb->serial);
        const char* firmware = udev_device_get_sysattr_value(dev, "bcdDevice");
532
        if(firmware)
533
534
            sscanf(firmware, "%hx", &kb->fwversion);
535
536
           kb \rightarrow fwversion = 0;
537
        int index = INDEX_OF(kb, keyboard);
540
        ckb_info("Connecting %s at %s%d\n", kb->name, devpath, index);
541
547
        const char* ep_str = udev_device_get_sysattr_value(dev, "bNumInterfaces");
548
        kb \rightarrow epcount = 0;
549
        if (ep str)
        sscanf(ep_str, "%d", &kb->epcount);
if(kb->epcount == 0){
550
552
            // This shouldn't happen, but if it does, assume EP count based on what the device is supposed to
553
            kb->epcount = (HAS FEATURES(kb, FEAT RGB) ? 4 : 3);
            ckb_warn("Unable to read endpoint count from udev, assuming %d...\n", kb->
554
      epcount);
555
556
        if (usbclaim(kb)) {
557
            ckb\_err("Failed to claim interfaces: %s\n", strerror(errno));
558
            return -1;
559
560
        return 0;
561 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.38.3.11 int os_usbrecv (usbdevice * kb, uchar * in_msg, const char * file, int line)

Parameters

kb	HE usbdevice*	
in_msg	ouffer to fill with the message received	
file	for debugging	
line	for debugging	

Returns

-1 on timeout, 0 on hard error, numer of bytes received otherwise

os_usbrecv does what its name says:

The comment at the beginning of the procedure causes the suspicion that the firmware versionspecific distinction is missing for receiving from usb endpoint 3 or 4. The commented code contains only the reception from EP4, but this may be wrong for a software version 2.0 or higher (see the code for os-usbsend ()).

So all the receiving is done via an ioctl() like in os_usbsend. The ioctl() is given a struct usbdevfs_ctrltransfer, in which the relevant parameters are entered:

bRequest- Type	bRequest	wValue	EP	size	Timeout	data
0xA1	0x01	0x0200	endpoint to be addressed from epcount - 1	MSG_SIZE	5ms	the message buffer pointer

Device to	1 =	specific	Interface #	64	5000	in_msg
Host,	RECEIVE?					
Type=Class,						
Recipi-						
ent=Interface						

The ioctl() returns the number of bytes received. Here is the usual check again:

- If the return value is -1 AND the error is a timeout (ETIMEOUT), os_usbrecv() will return -1 to indicate that it is probably a recoverable problem and a retry is recommended.
- For another negative value or other error identifier OR 0 bytes are received, 0 is returned as an identifier for a heavy error.
- · In all other cases, the function returns the number of bytes received.

If this is not the entire blocksize (MSG_SIZE bytes), an error message is issued on the standard error channel [warning "Read YY bytes (expected 64)"].

Definition at line 127 of file usb_linux.c.

References ckb_err_fn, ckb_warn_fn, usbdevice::epcount, usbdevice::handle, and MSG_SIZE.

Referenced by usbrecv().

```
127
128
129
        // This is what CUE does, but it doesn't seem to work on linux.
130
        /*if(kb->fwversion >= 0x130){
131
            struct usbdevfs bulktransfer transfer:
132
            memset(&transfer, 0, sizeof(transfer));
transfer.ep = 0x84;
133
            transfer.len = MSG_SIZE;
135
            transfer.timeout = 5000;
136
            transfer.data = in_msg;
137
            res = ioctl(kb->handle - 1, USBDEVFS_BULK, &transfer);
       } else {*/
138
            struct usbdevfs_ctrltransfer transfer = { 0xa1, 0x01, 0x0300, kb->
139
      epcount - 1, MSG_SIZE, 5000, in_msg };
            res = ioctl(kb->handle - 1, USBDEVFS_CONTROL, &transfer);
141
142
        if(res <= 0){
            ckb_err_fn("%s\n", file, line, res ? strerror(errno) : "No data read");
if(res == -1 && errno == ETIMEDOUT)
143
144
145
                return -1;
146
147
       } else if(res != MSG_SIZE)
148
            ckb_warn_fn("Read %d bytes (expected %d)\n", file, line, res,
149
      MSG_SIZE);
150 #ifdef DEBUG_USB_RECV
        char converted[MSG_SIZE*3 + 1];
152
        for(int i=0;i<MSG_SIZE;i++)</pre>
153
            sprintf(&converted[i*3], "%02x ", in_msg[i]);
154
        ckb_warn_fn("Recv %s\n", file, line, converted);
155 #endif
156
        return res;
157 }
```

Here is the caller graph for this function:



5.38.3.12 int os_usbsend (usbdevice * kb, const uchar * out_msg, int is_recv, const char * file, int line)

Parameters

kb	THE usbdevice*			
out_msg	he MSGSIZE char long buffer to send			
is_recv	if true, just send an ioctl for further reading packets. If false, send the data at out_msg .			
file	for debugging			
line	for debugging			

Returns

-1 on timeout (try again), 0 on hard error, numer of bytes sent otherwise

os_usbsend has two functions:

- if is_recv == false, it tries to send a given MSG_SIZE buffer via the usb interface given with kb.
- otherwise a request is sent via the usb device to initiate the receiving of a message from the remote device.

The functionality for sending distinguishes two cases, depending on the version number of the firmware of the connected device:

If the firmware is less or equal 1.2, the transmission is done via an ioctl(). The ioctl() is given a struct usbdevfs_ctrltransfer, in which the relevant parameters are entered:

bRequest- Type	bRequest	wValue	EP	size	Timeout	data
0x21	0x09	0x0200	endpoint / IF to be addressed from epcount-1	MSG_SIZE	5000 (=5ms)	the message buffer pointer
Host to Device, Type=Class, Recipi- ent=Interface	9 = Send data?	specific	last or pre-last device #	64	5000	out_msg

The ioctl command is USBDEVFS_CONTROL.

The same constellation is used if the device is requested to send its data (is recv = true).

For a more recent firmware and is_recv = false, the ioctl command USBDEVFS_CONTROL is not used (this tells the bus to enter the control mode), but the bulk method is used: USBDEVFS_BULK. This is astonishing, because all of the endpoints are type Interrupt, not bulk.

Anyhow, forthis purpose a different structure is used for the ioctl() (struct **usbdevfs_bulktransfer**) and this is also initialized differently:

The length and timeout parameters are given the same values as above. The formal parameter out_msg is also passed as a buffer pointer. For the endpoints, the firmware version is differentiated again:

For a firmware version between 1.3 and <2.0 endpoint 4 is used, otherwise (it can only be >=2.0) endpoint 3 is used.

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

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

The ioctl() - no matter what type - returns the number of bytes sent. Now comes the usual check:

• If the return value is -1 AND the error is a timeout (ETIMEOUT), os_usbsend() will return -1 to indicate that it is probably a recoverable problem and a retry is recommended.

• For another negative value or other error identifier OR 0 bytes sent, 0 is returned as a heavy error identifier.

• In all other cases, the function returns the number of bytes sent.

If this is not the entire blocksize (MSG_SIZE bytes), an error message is issued on the standard error channel [warning "Wrote YY bytes (expected 64)"].

If DEBUG_USB is set during compilation, the number of bytes sent and their representation are logged to the error channel.

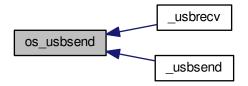
Definition at line 66 of file usb linux.c.

References ckb_err_fn, ckb_warn_fn, usbdevice::epcount, usbdevice::fwversion, usbdevice::handle, and MSG_SIZE.

Referenced by usbrecv(), and usbsend().

```
66
67
       int res:
       if (kb->fwversion >= 0x120 && !is_recv) {
68
           struct usbdevfs_bulktransfer transfer;
69
            memset(&transfer, 0, sizeof(transfer));
            transfer.ep = (kb->fwversion >= 0x130 && kb->fwversion < 0x200) ? 4 : 3;
transfer.len = MSG_SIZE;
71
72
73
            transfer.timeout = 5000;
            transfer.data = (void*)out_msg;
74
75
            res = ioctl(kb->handle - 1, USBDEVFS_BULK, &transfer);
76
       } else {
77
            struct usbdevfs_ctrltransfer transfer = { 0x21, 0x09, 0x0200, kb->
      epcount - 1, MSG_SIZE, 5000, (void*)out_msg );
    res = ioctl(kb->handle - 1, USBDEVFS_CONTROL, &transfer);
78
79
80
       if(res <= 0){
            ckb_err_fn("%s\n", file, line, res ? strerror(errno) : "No data written");
82
            if (res == -1 && errno == ETIMEDOUT)
83
                return -1;
84
            else
       return 0;
} else if(res != MSG_SIZE)
8.5
86
87
            ckb_warn_fn("Wrote %d bytes (expected %d)\n", file, line, res,
      MSG_SIZE);
88 #ifdef DEBUG_USB
       char converted[MSG_SIZE*3 + 1];
89
90
       for (int i=0; i < MSG_SIZE; i++)</pre>
            sprintf(&converted[i*3], "%02x ", out_msg[i]);
91
       ckb_warn_fn("Sent %s\n", file, line, converted);
93 #endif
94
       return res;
95 }
```

Here is the caller graph for this function:



5.38.3.13 const char* product_str (short product)

Parameters

product	is the short USB device product ID
---------	------------------------------------

Returns

string to identify a type of device (see below)

product str returns a condensed view on what type of device we have.

At present, various models and their properties are known from corsair products. Some models differ in principle (mice and keyboards), others differ in the way they function (for example, RGB and non RGB), but they are very similar

Here, only the first point is taken into consideration and we return a unified model string. If the model is not known with its number, *product_str* returns an empty string.

The model numbers and corresponding strings wwith the numbers in hex-string are defined in usb.h

At present, this function is used to initialize kb->name and to give information in debug strings.

Attention

The combinations below have to fit to the combinations in the macros mentioned above. So if you add a device with a new number, change both.

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

Definition at line 70 of file usb.c.

References P_HARPOON, P_K65, P_K65_LUX, P_K65_NRGB, P_K65_RFIRE, P_K70, P_K70_LUX, P_K70_-LUX_NRGB, P_K70_NRGB, P_K70_RFIRE, P_K70_RFIRE_NRGB, P_K95, P_K95_NRGB, P_K95_PLATINUM, P_M65, P_M65_PRO, P_SABRE_L, P_SABRE_N, P_SABRE_O, P_SABRE_O2, P_SCIMITAR, P_SCIMITAR_P-RO, P_STRAFE, and P_STRAFE_NRGB.

Referenced by _mkdevpath(), and _setupusb().

```
if(product == P_K95 || product == P_K95_NRGB || product ==
      P_K95_PLATINUM)
            return "k95";
72
       if(product == P_K70 || product == P_K70_NRGB || product ==
7.3
      P_K70_LUX || product == P_K70_LUX_NRGB || product == P_K70_RFIRE || product == P_K70_RFIRE_NRGB)
75
       if(product == P_K65 || product == P_K65_NRGB || product ==
      P_K65_LUX || product == P_K65_RFIRE)
       return "k65";
if(product == P_STRAFE || product == P_STRAFE_NRGB)
76
77
            return "strafe";
78
       if (product == P_M65 || product == P_M65_PRO)
            return "m65";
       if(product == P_SABRE_O || product == P_SABRE_L || product ==
81
      P_SABRE_N || product == P_SABRE_O2 || product == P_HARPOON)
    return "sabre";
82
       if(product == P_SCIMITAR || product == P_SCIMITAR_PRO)
83
            return "scimitar";
       return "";
86 }
```

Here is the caller graph for this function:



5.38.3.14 int revertusb (usbdevice * kb)

Parameters

kb	THE usbdevice*

Returns

0 on success or if device needs firmware upgrade, -1 otherwise

revertusb sets a given device to inactive (hardware controlled) mode if not a fw-ugrade is indicated First is checked, whether a firmware-upgrade is indicated for the device. If so, revertusb() returns 0.

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

Anyway, the following steps are similar to some other procs, dealing with low level usb handling:

• If we do not have an RGB device, a simple setting to Hardware-mode (NK95_HWON) is sent to the device via n95cmd().

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

• If we have an RGB device, setactive() is called with second param active = false. That function will have a look on differences between keyboards and mice.

More precisely setactive() is just a macro to call via the kb->vtable enties either the active() or the idle() function where the vtable points to. setactive() may return error indications. If so, revertusb() returns -1, otherwise 0 in any other case.

Definition at line 407 of file usb.c.

References FEAT_RGB, HAS_FEATURES, NEEDS_FW_UPDATE, NK95_HWON, nk95cmd, and setactive.

Referenced by quitWithLock().

```
407
408
        if (NEEDS_FW_UPDATE(kb))
409
             return 0:
        if (!HAS_FEATURES(kb, FEAT_RGB)) {
410
411
             nk95cmd(kb, NK95_HWON);
412
             return 0;
413
414
        if (setactive(kb, 0))
415
             return -1;
        return 0;
416
417 }
```

Here is the caller graph for this function:



5.38.3.15 void setupusb (usbdevice * kb)

Attention

Lock a device's dmutex (see device.h) before accessing the USB interface.

Parameters

kb	THE usbdevice* used everywhere	
OUT]	kb->thread is used to store the thread ID of the fresh created thread.	

setupusb starts a thread with kb as parameter and _setupusb() as entrypoint.

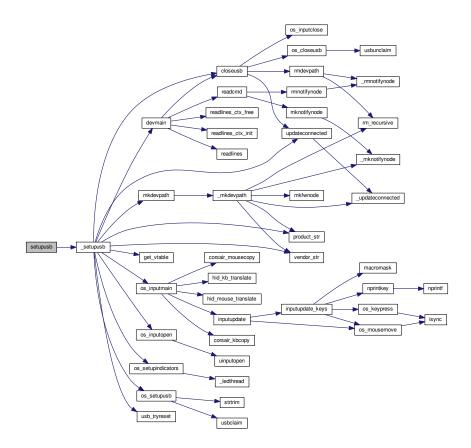
Set up a USB device after its handle is open. Spawns a new thread _setupusb() with standard parameter kb. dmutex must be locked prior to calling this function. The function will unlock it when finished. In kb->thread the thread id is mentioned, because closeusb() needs this info for joining that thread again.

Definition at line 386 of file usb.c.

References _setupusb(), ckb_err, imutex, and usbdevice::thread.

Referenced by usbadd().

Here is the call graph for this function:



Here is the caller graph for this function:



5.38.3.16 int usb_tryreset (usbdevice * kb)

Parameters

in,out	kb	THE usbdevice*
in	reset_stop	global variable is read

Returns

0 on success, -1 otherwise

usb_tryreset does what the name means: Try to reset the usb via resetusb()

This function is called if an usb command ran into an error in case of one of the following two situations:

· When setting up a new usb device and the start() function got an error (

See Also

```
_setupusb())
```

· If upgrading to a new firmware gets an error (

See Also

```
cmd fwupdate()).
```

The previous action which got the error will NOT be re-attempted.

In an endless loop usb_tryreset() tries to reset the given usb device via the macro resetusb().

This macro calls resetusb() with debugging information.

_resetusb() sends a command via the operating system dependent function os_resetusb() and - if successful - reinitializes the device. os_resetusb() returns -2 to indicate a broken device and all structures should be removed for it.

In that case, the loop is terminated, an error message is produced and usb tryreset() returns -1.

In case resetusb() has success, the endless loop is left via a return 0 (success).

If the return value from resetusb() is -1, the loop is continued with the next try.

If the global variable **reset_stop** is set directly when the function is called or after each try, **usb_tryreset()** stops working and returns -1.

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

Definition at line 465 of file usb.c.

References ckb_err, ckb_info, reset_stop, and resetusb.

Referenced by setupusb(), and cmd fwupdate().

```
465
466
        if (reset stop)
467
            return -1;
        ckb_info("Attempting reset...\n");
469
        while(1){
            int res = resetusb(kb);
470
471
            if(!res){
                ckb_info("Reset success\n");
472
                return 0;
474
475
            if(res == -2 || reset_stop)
476
                break;
477
478
        ckb_err("Reset failed. Disconnecting.\n");
479
        return -1;
480 }
```

Here is the caller graph for this function:



```
5.38.3.17 void usbkill ( )
```

Definition at line 803 of file usb_linux.c.

Referenced by quitWithLock().

```
803 {
804 udev_unref(udev);
805 udev = 0;
806 }
```

Here is the caller graph for this function:



```
5.38.3.18 int usbmain ( )
```

Start the USB main loop. Returns program exit code when finished. usbmain is called by main() after setting up all other stuff.

Returns

0 normally or -1 if fatal error occurs (up to now only if no new devices are available)

First check whether the uinput module is loaded by the kernel.

Todo Why isn't missing of uinput a fatal error?

Create the udev object with udev_new() (is a function from libudev.h) terminate -1 if error Enumerate all currently connected devices

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

Definition at line 743 of file usb_linux.c.

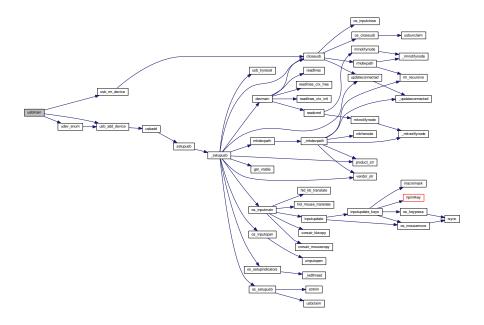
References ckb_fatal, ckb_warn, udev_enum(), usb_add_device(), and usb_rm_device(). Referenced by main().

```
743 {
748    // Load the uinput module (if it's not loaded already)
749    if(system("modprobe uinput") != 0)
```

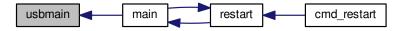
```
750
               ckb_warn("Failed to load uinput module\n");
751
755
          if(!(udev = udev_new())) {
               {\tt ckb\_fatal("Failed \ to \ initialize \ udev \ in \ usbmain(), \ usb\_linux.c\n");}
756
757
               return -1;
758
759
762
         udev_enum();
763
         // Done scanning. Enter a loop to poll for device updates
struct udev_monitor* monitor = udev_monitor_new_from_netlink(udev, "udev");
udev_monitor_filter_add_match_subsystem_devtype(monitor, "usb", 0);
766
767
768
         udev_monitor_enable_receiving(monitor);
769
770
          // Get an fd for the monitor
771
          int fd = udev_monitor_get_fd(monitor);
772
773
          fd_set fds;
         while (udev) {
    FD_ZERO(&fds);
774
775
               FD_SET(fd, &fds);
776
                  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);
777
778
779
                    if(!dev)
780
                         continue;
781
                    const char* action = udev_device_get_action(dev);
782
                    if(!action){
783
                         udev_device_unref(dev);
784
                         continue;
785
786
                    // Add/remove device
                    if(!strcmp(action, "add")){
787
                         int res = usb_add_device(dev);
if(res == 0)
788
789
790
                              continue;
                         // If the device matched but the handle wasn't opened correctly, re-enumerate (this
791
        sometimes solves the problem)

if (res == -1)
792
793
                             udev_enum();
794
                    } else if(!strcmp(action, "remove"))
795
                        usb_rm_device(dev);
796
                    udev_device_unref(dev);
797
               }
798
799
         udev_monitor_unref(monitor);
800
         return 0;
801 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.38.3.19 const char* vendor_str (short vendor)

vendor_str Vendor/product string representations

Parameters

	ndor ID
--	---------

Returns

```
a string: either "" or "corsair"
```

uncomment the following Define to see USB packets sent to the device

vendor_str returns "corsair" iff the given vendor argument is equal to V_CORSAIR (0x1bc) else it returns ""

Attention

There is also a string defined V_CORSAIR_STR, which returns the device number as string in hex "1b1c".

Definition at line 43 of file usb.c.

References V_CORSAIR.

Referenced by _mkdevpath(), and _setupusb().

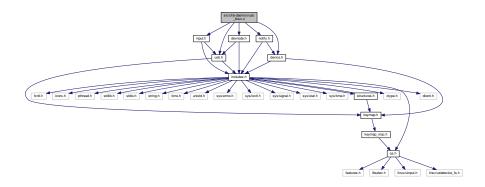
Here is the caller graph for this function:



5.39 src/ckb-daemon/usb_linux.c File Reference

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

Include dependency graph for usb_linux.c:



Data Structures

· struct _model

Macros

- #define TEST RESET(op)
 - TEST_RESET doesa "try / catch" for resetting the usb interface.
- #define N_MODELS (sizeof(models) / sizeof(_model))

Functions

- int os_usbsend (usbdevice *kb, const uchar *out_msg, int is_recv, const char *file, int line)
 - os_usbsend sends a data packet (MSG_SIZE = 64) Bytes long
- int os_usbrecv (usbdevice *kb, uchar *in_msg, const char *file, int line)
 - os_usbrecv receives a max MSGSIZE long buffer from usb device
- int _nk95cmd (usbdevice *kb, uchar bRequest, ushort wValue, const char *file, int line)
 - _nk95cmd If we control a non RGB keyboard, set the keyboard via ioctl with usbdevfs_ctrltransfer
- void os_sendindicators (usbdevice *kb)
- void * os_inputmain (void *context)
 - os_inputmain This function is run in a separate thread and will be detached from the main thread, so it needs to clean up its own resources.
- static int usbunclaim (usbdevice *kb, int resetting)
- void os_closeusb (usbdevice *kb)
- static int usbclaim (usbdevice *kb)
- int os_resetusb (usbdevice *kb, const char *file, int line)
- void strtrim (char *string)
- int os_setupusb (usbdevice *kb)
- int usbadd (struct udev_device *dev, short vendor, short product)
- static int usb add device (struct udev device *dev)
 - Add a udev device. Returns 0 if device was recognized/added.
- static void usb_rm_device (struct udev_device *dev)
 - usb_rm_device find the usb port to remove and close it via closeusb().
- static void udev_enum ()
 - udev_enum use the udev_enumerate_add_match_subsystem() to get all you need but only that.
- int usbmain ()
- · void usbkill ()

Stop the USB system.

Variables

• static char kbsyspath [9][FILENAME_MAX]

all open usb devices have their system path names here in this array.

- static struct udev * udev
- pthread_t usbthread

struct udef is defined in /usr/include/libudev.h

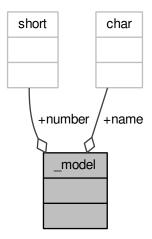
- pthread_t udevthread
- static _model models []

5.39.1 Data Structure Documentation

5.39.1.1 struct _model

Definition at line 612 of file usb_linux.c.

Collaboration diagram for _model:



Data Fields

const char *	name	
short	number	

5.39.2 Macro Definition Documentation

5.39.2.1 #define N_MODELS (sizeof(models) / sizeof(_model))

Definition at line 650 of file usb_linux.c.

Referenced by usb_add_device().

5.39.2.2 #define TEST_RESET(op)

Value:

Definition at line 462 of file usb_linux.c.

Referenced by os_resetusb().

5.39.3 Function Documentation

5.39.3.1 int_nk95cmd (usbdevice * kb, uchar bRequest, ushort wValue, const char * file, int line)

To send control packets to a non RGB non color K95 Keyboard, use this function. Normally it is called via the nk95cmd() macro.

If it is the wrong device for which the function is called, 0 is returned and nothing done. Otherwise a usbdevfs_ctrltransfer structure is filled and an USBDEVFS_CONTROL ioctl() called.

bRequest-	bRequest	wValue	EP	size	Timeout	data
Туре						
0x40	see table below to switch hardware- modus at Keyboard	wValue	device	MSG_SIZE	5ms	the message buffer pointer
Host to Device, Type=Vendor, Recipi- ent=Device	bRequest parameter	given wValue Parameter	device 0	0 data to write	5000	null

If a 0 or a negative error number is returned by the ioctl, an error message is shown depending on the error or "No data written" if retval was 0. In either case 1 is returned to indicate the error. If the ioctl returned a value > 0, 0 is returned to indicate no error.

Currently the following combinations for bRequest and wValue are used:

Device	what it might to	constant	bRequest	wValue
	do			
non RGB Keyboard	set HW-modus on	HWON	0x0002	0x0030
	(leave the ckb			
	driver)			
non RGB Keyboard	set HW-modus off	HWOFF	0x0002	0x0001
	(initialize the ckb			
	driver)			
non RGB Keyboard	set light modus M1	NK95_M1	0x0014	0x0001
	in single-color			
	keyboards			
non RGB Keyboard	set light modus M2	NK95_M2	0x0014	0x0002
	in single-color			
	keyboards			
non RGB Keyboard	set light modus M3	NK95_M3	0x0014	0x0003
	in single-color			
	keyboards			

See Also

usb.h

Definition at line 187 of file usb linux.c.

References ckb_err_fn, usbdevice::handle, P_K95_NRGB, and usbdevice::product.

```
187
188
        if(kb->product != P_K95_NRGB)
189
            return 0;
190
        struct usbdevfs_ctrltransfer transfer = { 0x40, bRequest, wValue, 0, 0, 5000, 0 };
        int res = ioctl(kb->handle - 1, USBDEVFS_CONTROL, &transfer);
191
            ckb_err_fn("%s\n", file, line, res ? strerror(errno) : "No data written");
return 1;
192
        <u>if</u>(res <= 0){
193
194
195
196
        return 0;
197 }
```

```
5.39.3.2 void os_closeusb ( usbdevice * kb )
```

os_closeusb unclaim it, destroy the udev device and clear data structures at kb

os_closeusb is the linux specific implementation for closing an active usb port.

If a valid handle is given in the kb structure, the usb port is unclaimed (usbunclaim()).

The device in unrefenced via library function udev_device_unref().

handle, udev and the first char of kbsyspath are cleared to 0 (empty string for kbsyspath).

Definition at line 422 of file usb linux.c.

References usbdevice::handle, INDEX_OF, kbsyspath, keyboard, usbdevice::udev, and usbunclaim().

Referenced by closeusb().

```
423
        if (kb->handle) {
424
            usbunclaim(kb, 0);
425
            close(kb->handle - 1);
426
427
        if (kb->udev)
428
            udev_device_unref(kb->udev);
429
        kb \rightarrow handle = 0;
430
        kb->udev = 0;
431
        kbsyspath[INDEX_OF(kb, keyboard)][0] = 0;
432 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.39.3.3 void* os_inputmain (void * context)

os_inputmain is run in a separate thread and will be detached from the main thread, so it needs to clean up its own resources.

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

- 1. set up an URB (Userspace Ressource Buffer) to communicate with the USBDEVFS * ioctl()s
- 2. perform the ioctl()
- 3. interpretate the information got into the URB buffer or handle error situations and retry operation or leave the endless loop
- 4. inform the os about the data
- 5. loop endless via 2.
- 6. if endless loop has gone, deinitalize the interface, free buffers etc.
- 7. return null

Here the actions in detail:

Monitor input transfers on all endpoints for non-RGB devices For RGB, monitor all but the last, as it's used for input/output

Get an usbdevfs_urb data structure and clear it via memset()

Hopefully the buffer lengths are equal for all devices with congruent types. You can find out the correctness for your device with Isusb –v or similar on macOS. Currently the following combinations are known and implemented:

device	detect with macro combination	endpoint #	buffer-length
each	none	0	8
RGB Mouse	IS_RGB && IS_MOUSE	1	10
RGB Keyboard	IS_RGB && !IS_MOUSE	1	21
RGB Mouse or Keyboard	IS_RGB	2	MSG_SIZE (64)
non RGB Mouse or	!IS_RGB	1	4
Keyboard			
non RGB Mouse or	!IS_RGB	2	15
Keyboard			

Now submit all the URBs via ioctl(USBDEVFS_SUBMITURB) with type USBDEVFS_URB_TYPE_INTERRUPT (the endpoints are defined as type interrupt). Endpoint number is 0x80..0x82 or 0x83, depending on the model.

The userSpaceFS knows the URBs now, so start monitoring input

if the ioctl returns something != 0, let's have a deeper look what happened. Broken devices or shutting down the entire system leads to closing the device and finishing this thread.

If just an EPIPE ocurred, give the device a CLEAR_HALT and resubmit the URB.

A correct REAPURB returns a Pointer to the URB which we now have a closer look into. Lock all following actions with imutex.

Process the input depending on type of device. Interprete the actual size of the URB buffer

device	detect with macro combination	seems to be endpoint #	actual buffer-length	function called
mouse (RGB and non RGB)	IS_MOUSE	nA	8, 10 or 11	hid_mouse translate()
mouse (RGB and non RGB)	IS_MOUSE	nA	MSG_SIZE (64)	corsair mousecopy()
RGB Keyboard	IS_RGB && !IS_MOUSE	1	8 (BIOS Mode)	hid_kb_translate()
RGB Keyboard	IS_RGB && !IS_MOUSE	2	5 or 21, KB inactive!	hid_kb_translate()
RGB Keyboard	IS_RGB && !IS_MOUSE	3?	MSG_SIZE	corsair_kbcopy()
non RGB Keyboard	!IS_RGB && !IS_MOUSE	nA	nA	hid_kb_translate()

The input data is transformed and copied to the kb structure. Now give it to the OS and unlock the imutex afterwards.

Re-submit the URB for the next run.

If the endless loop is terminated, clean up by discarding the URBs via ioctl(USBDEVFS_DISCARDURB), free the URB buffers and return a null pointer as thread exit code.

Definition at line 232 of file usb linux.c.

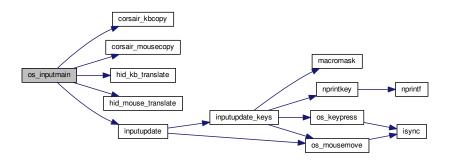
References usbdevice::active, ckb_info, corsair_kbcopy(), corsair_mousecopy(), devpath, usbdevice::epcount, usbdevice::handle, hid_kb_translate(), hid_mouse_translate(), imutex, INDEX_OF, usbdevice::input, inputupdate(), IS_MOUSE, IS_RGB, keyboard, usbinput::keys, MSG_SIZE, usbdevice::product, usbinput::rel_x, usbinput::rel_y, and usbdevice::vendor.

Referenced by _setupusb().

```
232
                                       {
233
        usbdevice* kb = context;
234
        int fd = kb->handle - 1;
235
        short vendor = kb->vendor, product = kb->product;
236
        int index = INDEX_OF(kb, keyboard);
237
        ckb_info("Starting input thread for %s%d\n", devpath, index);
238
        int urbcount = IS_RGB(vendor, product) ? (kb->epcount - 1) : kb->
243
      epcount;
244
246
        struct usbdevfs_urb urbs[urbcount];
247
        memset(urbs, 0, sizeof(urbs));
248
262
        urbs[0].buffer_length = 8;
263
        if(IS_RGB(vendor, product)){
264
            if(IS_MOUSE(vendor, product))
265
                urbs[1].buffer_length = 10;
266
            else
                urbs[1].buffer length = 21;
267
268
            urbs[2].buffer_length = MSG_SIZE;
            if(urbcount != 3)
269
270
                urbs[urbcount - 1].buffer_length = MSG_SIZE;
271
            urbs[1].buffer_length = 4;
urbs[2].buffer_length = 15;
272
273
274
275
        for(int i = 0; i < urbcount; i++) {</pre>
278
279
            urbs[i].type = USBDEVFS_URB_TYPE_INTERRUPT;
280
            urbs[i].endpoint = 0x80 | (i + 1);
            urbs[i].buffer = malloc(urbs[i].buffer_length);
281
            ioctl(fd, USBDEVFS_SUBMITURB, urbs + i);
282
283
        }
284
286
        while (1) {
287
            struct usbdevfs_urb* urb = 0;
288
291
            if (ioctl(fd, USBDEVFS REAPURB, &urb)) {
                if (errno == ENODEV || errno == ENOENT || errno == ESHUTDOWN)
292
293
                     // Stop the thread if the handle closes
```

```
294
                    break;
295
                else if(errno == EPIPE && urb){
297
                    ioctl(fd, USBDEVFS_CLEAR_HALT, &urb->endpoint);
298
                    // Re-submit the URB \,
299
                    if(urb)
300
                        ioctl(fd, USBDEVFS_SUBMITURB, urb);
                    urb = 0;
301
302
                }
303
            }
304
            if (urb) {
308
                pthread_mutex_lock(imutex(kb));
320
                if(Is_MOUSE(vendor, product)){
    switch(urb->actual_length){
321
322
323
                     case 8:
324
                     case 10:
325
                    case 11:
326
                        // HID mouse input
                        hid_mouse_translate(kb->input.keys, &kb->
327
      input.rel_x, &kb->input.rel_y, -(urb->endpoint & 0xF), urb->actual_length, urb->buffer)
328
                        break;
                    case MSG_SIZE:
329
                        // Corsair mouse input
330
331
                        corsair_mousecopy(kb->input.keys, -(urb->endpoint & 0xF), urb
      ->buffer);
332
333
                } else if(IS_RGB(vendor, product)){
334
335
                    switch(urb->actual_length) {
336
                    case 8:
337
                         // RGB EP 1: 6KRO (BIOS mode) input
338
                        hid_kb_translate(kb->input.keys, -1, urb->actual_length, urb->
      buffer);
                        break;
339
                    case 21:
340
                    case 5:
341
342
                        // RGB EP 2: NKRO (non-BIOS) input. Accept only if keyboard is inactive
343
                        if(!kb->active)
344
                            hid_kb_translate(kb->input.keys, -2, urb->actual_length,
      urb->buffer);
345
                    break;
case MSG SIZE:
346
                        // RGB EP 3: Corsair input
347
                        corsair_kbcopy(kb->input.keys, -(urb->endpoint & 0xF), urb->
348
      buffer);
349
                        break:
350
                    }
351
                } else {
                    // Non-RGB input
352
                    hid_kb_translate(kb->input.keys, urb->endpoint & 0xF, urb->
353
      actual_length, urb->buffer);
354
357
                inputupdate(kb);
                pthread_mutex_unlock(imutex(kb));
358
                ioctl(fd, USBDEVFS_SUBMITURB, urb);
360
361
                urb = 0;
362
            }
363
        }
364
        368
369
370
371
            free(urbs[i].buffer);
372
373
        return 0;
374 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.39.3.4 int os_resetusb (usbdevice * kb, const char * file, int line)

os_resetusb is the os specific implementation for resetting usb

Try to reset an usb device in a linux user space driver.

- 1. unclaim the device, but do not reconnect the system driver (second param resetting = true)
- 2. reset the device via USBDEVFS_RESET command
- 3. claim the device again. Returns 0 on success, -2 if device should be removed and -1 if reset should by tried again

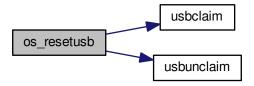
Todo it seems that no one wants to try the reset again. But I'v seen it somewhere...

Definition at line 480 of file usb_linux.c.

References usbdevice::handle, TEST_RESET, usbclaim(), and usbunclaim().

Referenced by _resetusb().

Here is the call graph for this function:



Here is the caller graph for this function:



5.39.3.5 void os_sendindicators (usbdevice * kb)

os_sendindicators update the indicators for the special keys (Numlock, Capslock and what else?)

os_sendindicators update the indicators for the special keys (Numlock, Capslock and what else?)

Read the data from kb->ileds ans send them via ioctl() to the keyboard.

bRequest- Type	bRequest	wValue	EP	size	Timeout	data
0x21	0x09	0x0200	Interface 0	MSG_SIZE 1 Byte	timeout 0,5ms	the message buffer pointer
Host to Device, Type=Class, Recipi- ent=Interface (why not endpoint?)	9 = SEND?	specific	0	1	500	struct* kb->ileds

The ioctl command is USBDEVFS_CONTROL.

Definition at line 212 of file usb linux.c.

References ckb_err, usbdevice::handle, and usbdevice::ileds.

Referenced by updateindicators_kb().

Here is the caller graph for this function:



5.39.3.6 int os_setupusb (usbdevice * kb)

os setupusb OS-specific setup for a specific usb device.

Perform the operating system-specific opening of the interface in os_setupusb(). As a result, some parameters should be set in kb (name, serial, fwversion, epcount = number of usb endpoints), and all endpoints should be claimed with usbclaim(). Claiming is the only point where os_setupusb() can produce an error (-1).

- · Copy device description and serial
- Copy firmware version (needed to determine USB protocol)
- · Do some output abaout connecting interfaces
- · Claim the USB interfaces

Todo in these modules a pullrequest is outstanding

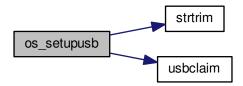
Definition at line 518 of file usb_linux.c.

References ckb_err, ckb_info, ckb_warn, devpath, usbdevice::epcount, FEAT_RGB, usbdevice::fwversion, HAS_-FEATURES, INDEX_OF, KB_NAME_LEN, keyboard, usbdevice::name, usbdevice::serial, SERIAL_LEN, strtrim(), usbdevice::udev, and usbclaim().

Referenced by setupusb().

```
518
521
        struct udev_device* dev = kb->udev;
522
        const char* name = udev_device_get_sysattr_value(dev, "product");
523
        if (name)
524
           strncpy(kb->name, name, KB_NAME_LEN);
        strtrim(kb->name);
525
526
        const char* serial = udev_device_get_sysattr_value(dev, "serial");
527
        if(serial)
528
            strncpy(kb->serial, serial, SERIAL_LEN);
529
        strtrim(kb->serial);
        const char* firmware = udev_device_get_sysattr_value(dev, "bcdDevice");
532
533
        if(firmware)
534
            sscanf(firmware, "%hx", &kb->fwversion);
535
           kb->fwversion = 0:
536
537
        int index = INDEX OF(kb, keyboard);
540
        ckb_info("Connecting %s at %s%d\n", kb->name, devpath, index);
541
547
        const char* ep_str = udev_device_get_sysattr_value(dev, "bNumInterfaces");
548
        kb->epcount = 0;
549
        if (ep_str)
        sscanf(ep_str, "%d", &kb->epcount);
if(kb->epcount == 0){
550
551
            // This shouldn't happen, but if it does, assume EP count based on what the device is supposed to
553
            kb->epcount = (HAS_FEATURES(kb, FEAT_RGB) ? 4 : 3);
554
            ckb_warn("Unable to read endpoint count from udev, assuming %d...\n", kb->
      epcount);
555
556
        if (usbclaim(kb)) {
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.39.3.7 int os_usbrecv (usbdevice * kb, uchar * in_msg, const char * file, int line)

os_usbrecv does what its name says:

The comment at the beginning of the procedure causes the suspicion that the firmware versionspecific distinction is missing for receiving from usb endpoint 3 or 4. The commented code contains only the reception from EP4, but this may be wrong for a software version 2.0 or higher (see the code for os-usbsend ()).

So all the receiving is done via an ioctl() like in os_usbsend. The ioctl() is given a struct usbdevfs_ctrltransfer, in which the relevant parameters are entered:

bRequest- Type	bRequest	wValue	EP	size	Timeout	data
0xA1	0x01	0x0200	endpoint to be addressed from epcount - 1	MSG_SIZE	5ms	the message buffer pointer
Device to Host, Type=Class, Recipi- ent=Interface	1 = RECEIVE?	specific	Interface #	64	5000	in_msg

The ioctl() returns the number of bytes received. Here is the usual check again:

- If the return value is -1 AND the error is a timeout (ETIMEOUT), os_usbrecv() will return -1 to indicate that it is probably a recoverable problem and a retry is recommended.
- For another negative value or other error identifier OR 0 bytes are received, 0 is returned as an identifier for a heavy error.

In all other cases, the function returns the number of bytes received.

If this is not the entire blocksize (MSG_SIZE bytes), an error message is issued on the standard error channel [warning "Read YY bytes (expected 64)"].

Definition at line 127 of file usb linux.c.

References ckb_err_fn, ckb_warn_fn, usbdevice::epcount, usbdevice::handle, and MSG_SIZE.

Referenced by _usbrecv().

```
127
        int res;
129
         // This is what CUE does, but it doesn't seem to work on linux.
130
        /*if(kb->fwversion >= 0x130){
             struct usbdevfs_bulktransfer transfer;
131
132
             memset(&transfer, 0, sizeof(transfer));
133
             transfer.ep = 0x84;
134
             transfer.len = MSG_SIZE;
135
             transfer.timeout = 5000;
136
            transfer.data = in_msg;
             res = ioctl(kb->handle - 1, USBDEVFS_BULK, &transfer);
137
        } else {*/
138
            struct usbdevfs_ctrltransfer transfer = { 0xa1, 0x01, 0x0300, kb-> nt - 1, MSG_SIZE, 5000, in_msg };
139
140
            res = ioctl(kb->handle - 1, USBDEVFS_CONTROL, &transfer);
141
142
        if(res <= 0){
             ckb_err_fn("%s\n", file, line, res ? strerror(errno) : "No data read");
if(res == -1 && errno == ETIMEDOUT)
143
144
145
                 return -1;
       return 0;
} else if(res != MSG_SIZE)
147
148
             ckb_warn_fn("Read %d bytes (expected %d)\n", file, line, res,
149
      MSG_SIZE);
150 #ifdef DEBUG_USB_RECV
        char converted[MSG_SIZE*3 + 1];
151
152
        for(int i=0;i<MSG_SIZE;i++)</pre>
             sprintf(&converted[i*3], "%02x ", in_msg[i]);
153
        ckb_warn_fn("Recv %s\n", file, line, converted);
154
155 #endif
156
        return res;
```

Here is the caller graph for this function:



5.39.3.8 int os_usbsend (usbdevice * kb, const uchar * out_msg, int is_recv, const char * file, int line)

os_usbsend has two functions:

- if is_recv == false, it tries to send a given MSG_SIZE buffer via the usb interface given with kb.
- otherwise a request is sent via the usb device to initiate the receiving of a message from the remote device.

The functionality for sending distinguishes two cases, depending on the version number of the firmware of the connected device:

If the firmware is less or equal 1.2, the transmission is done via an ioctl(). The ioctl() is given a struct usbdevfs_ctrltransfer, in which the relevant parameters are entered:

bRequest- Type	bRequest	wValue	EP	size	Timeout	data
0x21	0x09	0x0200	endpoint / IF to be addressed from epcount-1	MSG_SIZE	5000 (=5ms)	the message buffer pointer
Host to Device, Type=Class, Recipi- ent=Interface	9 = Send data?	specific	last or pre-last device #	64	5000	out_msg

The ioctl command is USBDEVFS CONTROL.

The same constellation is used if the device is requested to send its data (is_recv = true).

For a more recent firmware and is_recv = false, the ioctl command USBDEVFS_CONTROL is not used (this tells the bus to enter the control mode), but the bulk method is used: USBDEVFS_BULK. This is astonishing, because all of the endpoints are type Interrupt, not bulk.

Anyhow, forthis purpose a different structure is used for the ioctl() (struct **usbdevfs_bulktransfer**) and this is also initialized differently:

The length and timeout parameters are given the same values as above. The formal parameter out_msg is also passed as a buffer pointer. For the endpoints, the firmware version is differentiated again:

For a firmware version between 1.3 and <2.0 endpoint 4 is used, otherwise (it can only be >=2.0) endpoint 3 is used

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

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

The ioctl() - no matter what type - returns the number of bytes sent. Now comes the usual check:

- If the return value is -1 AND the error is a timeout (ETIMEOUT), os_usbsend() will return -1 to indicate that it is probably a recoverable problem and a retry is recommended.
- For another negative value or other error identifier OR 0 bytes sent, 0 is returned as a heavy error identifier.
- In all other cases, the function returns the number of bytes sent.

If this is not the entire blocksize (MSG_SIZE bytes), an error message is issued on the standard error channel [warning "Wrote YY bytes (expected 64)"].

If DEBUG_USB is set during compilation, the number of bytes sent and their representation are logged to the error channel.

Definition at line 66 of file usb linux.c.

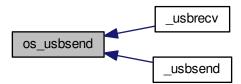
References ckb_err_fn, ckb_warn_fn, usbdevice::epcount, usbdevice::fwversion, usbdevice::handle, and MSG_SI-7F

Referenced by _usbrecv(), and _usbsend().

```
fed
int res;
if(kb->fwversion >= 0x120 && !is_recv) {
    struct usbdevfs_bulktransfer transfer;
    memset(&transfer, 0, sizeof(transfer));
    transfer.ep = (kb->fwversion >= 0x130 && kb->fwversion < 0x200) ? 4 : 3;
    transfer.len = MSG_SIZE;
itransfer.timeout = 5000;
transfer.data = (void*)out_msg;
}
</pre>
```

```
res = ioctl(kb->handle - 1, USBDEVFS_BULK, &transfer);
       } else {
           struct usbdevfs_ctrltransfer transfer = { 0x21, 0x09, 0x0200, kb->
77
      epcount - 1, MSG_SIZE, 5000, (void*)out_msg );
   res = ioctl(kb->handle - 1, USBDEVFS_CONTROL, &transfer);
78
79
80
       if(res <= 0){
81
           ckb_err_fn("%s\n", file, line, res ? strerror(errno) : "No data written");
82
            if (res == -1 && errno == ETIMEDOUT)
83
                return -1;
            else
84
       return 0;
} else if(res != MSG_SIZE)
85
86
           ckb_warn_fn("Wrote %d bytes (expected %d)\n", file, line, res,
      MSG_SIZE);
88 #ifdef DEBUG_USB
       char converted[MSG_SIZE*3 + 1];
89
       for(int i=0;i<MSG_SIZE;i++)</pre>
90
           sprintf(&converted[i*3], "%02x ", out_msg[i]);
91
       ckb_warn_fn("Sent %s\n", file, line, converted);
93 #endif
94
       return res;
95 }
```

Here is the caller graph for this function:



5.39.3.9 void strtrim (char * string)

strtrim trims a string by removing leading and trailing spaces.

Parameters

string

Definition at line 493 of file usb_linux.c.

Referenced by os_setupusb().

```
493
         // Find last non-space
494
         char* last = string;
for(char* c = string; *c != 0; c++){
495
496
497
              if(!isspace(*c))
498
                 last = c;
499
500
         last[1] = 0;
         // Find first non-space
501
         char* first = string;
for(; *first != 0; first++){
502
503
504
              if(!isspace(*first))
505
                  break;
506
507
         if(first != string)
508
             memmove(string, first, last - first);
509 }
```

Here is the caller graph for this function:



```
5.39.3.10 static void udev_enum( ) [static]
```

Reduce the hits of the enumeration by limiting to usb as technology and corsair as idVendor. Then filter with udev_enumerate_scan_devices () all hits.

The following call to udev_enumerate_get_list_entry() fetches the entire hitlist as udev_list_entry *.

Use udev_list_entry_foreach() to iterate through the hit set.

If both the device name exists (udev_list_entry_get_name) and the subsequent creation of a new udev_device (udev_device_new_from_syspath) is ok, the new device is added to the list with usb_add_device().

If the latter does not work, the new device is released again (udev_device_unref ()).

After the last iteration, the enumerator is released with udev_enumerate_unref ().

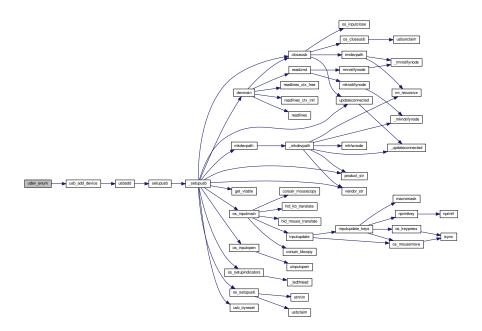
Definition at line 715 of file usb linux.c.

References usb_add_device(), and V_CORSAIR_STR.

Referenced by usbmain().

```
715
716
        struct udev_enumerate* enumerator = udev_enumerate_new(udev);
        udev_enumerate_add_match_subsystem(enumerator, "usb");
udev_enumerate_add_match_sysattr(enumerator, "idVendor", V_CORSAIR_STR);
718
719
        udev_enumerate_scan_devices(enumerator);
720
721
        struct udev_list_entry* devices, *dev_list_entry;
        devices = udev_enumerate_get_list_entry(enumerator);
722
723
        udev_list_entry_foreach(dev_list_entry, devices) {
724
            const char* path = udev_list_entry_get_name(dev_list_entry);
725
726
727
             struct udev_device* dev = udev_device_new_from_syspath(udev, path);
728
            if(!dev)
730
             // If the device matches a recognized device ID, open it
731
            if(usb_add_device(dev))
                 // Release device if not
732
733
                 udev_device_unref(dev);
734
735
        udev_enumerate_unref(enumerator);
736 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.39.3.11 static int usb_add_device (struct udev_device * dev) [static]

If the device id can be found, call usbadd() with the appropriate parameters.

Parameters

dev	the functions usb_*_device get a struct udev* with the neccessary hardware-related infor-
	mation.

Returns

the retval of usbadd() or 1 if either vendor is not corsair or product is not mentioned in model[].

First get the idVendor via udev_device_get_sysattr_value(). If this is equal to the ID-string of corsair ("1b1c"), get the idProduct on the same way.

If we can find the model name in the model array, call usbadd() with the model number.

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

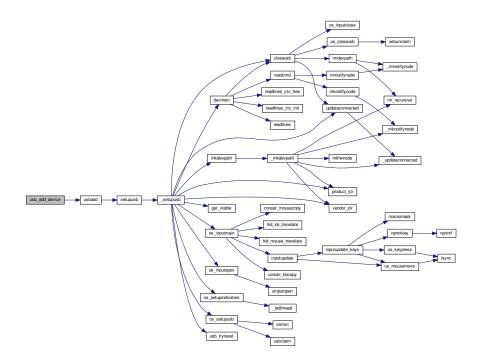
Definition at line 663 of file usb_linux.c.

References N_MODELS, usbadd(), V_CORSAIR, and V_CORSAIR_STR.

Referenced by udev_enum(), and usbmain().

```
663
           const char* vendor = udev_device_get_sysattr_value(dev, "idVendor");
if(vendor && !strcmp(vendor, V_CORSAIR_STR)) {
    const char* product = udev_device_get_sysattr_value(dev, "idProduct");
664
665
666
                 if(product){
667
668
                       for (_model* model = models; model < models +</pre>
        N_MODELS; model++) {
669
                             if(!strcmp(product, model->name)){
670
                                   return usbadd(dev, V_CORSAIR, model->number);
671
672
673
674
675
           return 1;
676 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.39.3.12 static void usb_rm_device (struct udev_device * dev) [static]

Parameters

dev	the functions usb_*_device get a struct udev* with the neccessary hardware-related infor-	1
	mation.	l

First try to find the system path of the device given in parameter dev. The index where the name is found is the same index we need to address the global keyboard array. That array holds all usbdevices.

Searching for the correct name in kbsyspath-array and closing the usb via closeusb() are protected by lock..unlock of the corresponding devmutex arraymember.

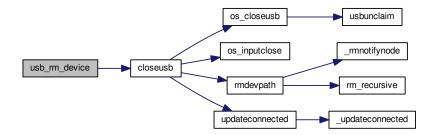
Definition at line 688 of file usb linux.c.

References closeusb(), DEV_MAX, devmutex, kbsyspath, and keyboard.

Referenced by usbmain().

```
688
689
          // Device removed. Look for it in our list of keyboards
690
          const char* syspath = udev_device_get_syspath(dev);
691
          if(!syspath || syspath[0] == 0)
692
          for(int i = 1; i < DEV_MAX; i++) {
    pthread_mutex_lock(devmutex + i);</pre>
693
694
               if(!strcmp(syspath, kbsyspath[i]))
    closeusb(keyboard + i);
695
696
697
              pthread_mutex_unlock(devmutex + i);
698
699 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.39.3.13 int usbadd (struct udev_device * dev, short vendor, short product)

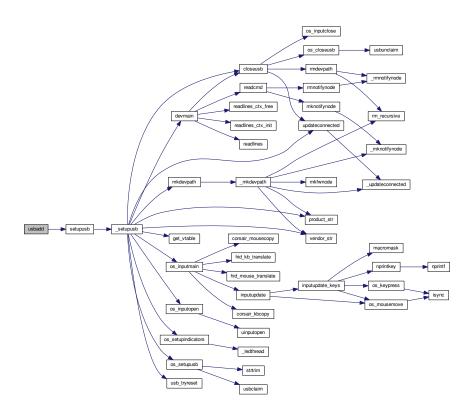
Definition at line 563 of file usb linux.c.

References ckb_err, DEV_MAX, dmutex, usbdevice::handle, IS_CONNECTED, kbsyspath, keyboard, usbdevice::product, setupusb(), usbdevice::udev, and usbdevice::vendor.

Referenced by usb_add_device().

```
563
                                                                                  {
564
         const char* path = udev_device_get_devnode(dev);
565
         const char* syspath = udev_device_get_syspath(dev);
         if(!path || !syspath || path[0] == 0 || syspath[0] == 0) {
   ckb_err("Failed to get device path\n");
566
567
568
              return -1;
569
570
         // Find a free USB slot
         for(int index = 1; index < DEV_MAX; index++) {
    usbdevice* kb = keyboard + index;</pre>
571
572
              if (pthread_mutex_trylock(dmutex(kb))) {
573
                  // If the mutex is locked then the device is obviously in use, so keep going
if(!strcmp(syspath, kbsyspath[index])){
574
575
576
                       // Make sure this existing keyboard doesn't have the same syspath (this shouldn't happen)
577
578
579
                  continue:
580
              if(!IS_CONNECTED(kb)){
581
582
                   // Open the sysfs device
583
                   kb->handle = open(path, O_RDWR) + 1;
584
                   if(kb->handle <= 0){</pre>
                       ckb_err("Failed to open USB device: %s\n", strerror(errno));
585
586
                       kb->handle = 0:
587
                       pthread_mutex_unlock(dmutex(kb));
588
                       return -1;
589
590
                       // Set up device
                       kb->udev = dev;
kb->vendor = vendor;
591
592
                       kb->product = product;
593
594
                       strncpy(kbsyspath[index], syspath, FILENAME_MAX);
595
                       // Mutex remains locked
596
                       setupusb(kb);
597
                       return 0;
598
599
600
             pthread_mutex_unlock(dmutex(kb));
601
602
         ckb_err("No free devices\n");
603
         return -1;
604 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.39.3.14 static int usbclaim (usbdevice * kb) [static]

usbclaim does claiming all EPs for the usb device gicen by kb.

Parameters

1.1-	THE colores
kb	THE USDGEVICE*

Returns

0 on success, -1 otherwise.

Claim all endpoints for a given device (remeber the decrementing of the file descriptor) via ioctl(USBDEVFS_DISC-ONNECT) and ioctl(USBDEVFS_CLAIMINTERFACE).

Error handling is done for the ioctl(USBDEVFS_CLAIMINTERFACE) only. If this fails, now an error message is thrown and -1 is returned. Function is called in usb_linux.c only, so it is declared as static now.

Definition at line 446 of file usb_linux.c.

References usbdevice::epcount, and usbdevice::handle.

Referenced by os_resetusb(), and os_setupusb().

```
446
         int count = kb->epcount;
447
         for (int i = 0; i < count; i++) {
    struct usbdevfs_ioctl ctl = { i, USBDEVFS_DISCONNECT, 0 };</pre>
449
450
              ioctl(kb->handle - 1, USBDEVFS_IOCTL, &ctl);
451
              if (ioctl(kb->handle - 1, USBDEVFS_CLAIMINTERFACE, &i)){
452
453
                   return -1;
454
456
         return 0;
457 }
```

Here is the caller graph for this function:

```
usbotian extension of contraction of
```

5.39.3.15 void usbkill ()

Definition at line 803 of file usb_linux.c.

Referenced by quitWithLock().

```
803 {
804 udev_unref(udev);
805 udev = 0;
806 }
```

Here is the caller graph for this function:



```
5.39.3.16 int usbmain ( )
```

Start the USB main loop. Returns program exit code when finished.

usbmain is called by main() after setting up all other stuff.

Returns

0 normally or -1 if fatal error occurs (up to now only if no new devices are available)

First check whether the uinput module is loaded by the kernel.

Todo Why isn't missing of uinput a fatal error?

Create the udev object with udev_new() (is a function from libudev.h) terminate -1 if error Enumerate all currently connected devices

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

Definition at line 743 of file usb_linux.c.

References ckb_fatal, ckb_warn, udev_enum(), usb_add_device(), and usb_rm_device().

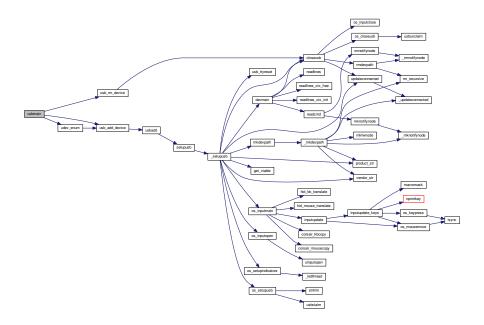
Referenced by main().

```
743
748
          // Load the uinput module (if it's not loaded already)
749
         if (system("modprobe uinput") != 0)
750
              ckb_warn("Failed to load uinput module\n");
751
         if(!(udev = udev_new())) {
755
756
              ckb_fatal("Failed to initialize udev in usbmain(), usb_linux.c\n");
757
              return -1;
758
759
762
         udev_enum();
763
         // Done scanning. Enter a loop to poll for device updates
struct udev_monitor* monitor = udev_monitor_new_from_netlink(udev, "udev");
766
767
768
         udev_monitor_filter_add_match_subsystem_devtype(monitor, "usb", 0);
769
         udev_monitor_enable_receiving(monitor);
         // Get an fd for the monitor
int fd = udev_monitor_get_fd(monitor);
770
771
772
         fd set fds;
773
         while (udev) {
774
              FD_ZERO(&fds);
775
776
              FD_SET(fd, &fds);
              \ensuremath{//} Block until an event is read
777
              if(select(fd + 1, &fds, 0, 0, 0) > 0 && FD_ISSET(fd, &fds)){
    struct udev_device* dev = udev_monitor_receive_device(monitor);
778
                   if(!dev)
780
781
                   const char* action = udev_device_get_action(dev);
782
                   if(!action){
783
                        udev_device_unref(dev);
784
                        continue:
785
786
                    // Add/remove device
```

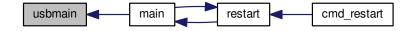
```
if(!strcmp(action, "add")){
                       int res = usb_add_device(dev);
if(res == 0)
788
789
                       continue;
// If the device matched but the handle wasn't opened correctly, re-enumerate (this
790
791
        sometimes solves the problem)

if (res == -1)
792
793
                            udev_enum();
794
                   } else if(!strcmp(action, "remove"))
                  usb_rm_device(dev);
udev_device_unref(dev);
795
796
797
798
799
         udev_monitor_unref(monitor);
800
         return 0;
801 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.39.3.17 static int usbunclaim (usbdevice * kb, int resetting) [static]

usbunclaim do an unclaiming of the usb device gicen by kb.

Parameters

kb	THE usbdevice*
resetting	boolean flag: If resseting is true, the caller will perform a bus reset command after unclaiming
	the device.

Returns

always 0.

Unclaim all endpoints for a given device (remeber the decrementing of the file descriptor) via ioctl(USBDEVFS_DI-SCARDURB).

Afterwards - if ressetting is false - do a USBDEVFS_CONNECT for EP 0 and 1. If it is a non RGB device, connect EP 2 also. The comment mentions RGB keyboards only, but as I understand the code, this is valid also for RGB mice

There is no error handling yet. Function is called in usb_linux.c only, so it is declared as static now.

Definition at line 393 of file usb_linux.c.

References usbdevice::epcount, FEAT_RGB, usbdevice::handle, and HAS_FEATURES.

Referenced by os closeusb(), and os resetusb().

```
393
394
         int handle = kb->handle - 1;
395
         int count = kb->epcount;
396
         for (int i = 0; i < count; i++) {</pre>
397
              ioctl(handle, USBDEVFS_RELEASEINTERFACE, &i);
398
        ^{\prime\prime} 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.
399
400
         // Reconnecting any of the others causes trouble.
401
         if (!resetting) {
402
              struct usbdevfs_ioctl ctl = { 0, USBDEVFS_CONNECT, 0 };
403
              ioctl(handle, USBDEVFS_IOCTL, &ctl);
404
              ctl.ifno = 1;
              ioctl(handle, USBDEVFS_IOCTL, &ctl);
// Also reconnect iface #2 (HID) for non-RGB keyboards
405
406
              if(!HAS_FEATURES(kb, FEAT_RGB)){
407
                  ctl.ifno = 2;
408
409
                  ioctl(handle, USBDEVFS_IOCTL, &ctl);
410
411
         return 0;
412
```

Here is the caller graph for this function:



5.39.4 Variable Documentation

5.39.4.1 char kbsyspath[9][FILENAME_MAX] [static]

Definition at line 11 of file usb_linux.c.

Referenced by os_closeusb(), usb_rm_device(), and usbadd().

5.39.4.2 _model models[] [static]

Initial value:

340 File Documentation

```
= {
            "1b17",
"1b07",
"1b37",
"1b39",
"1b13",
"1b38",
"1b38",
"1b38",
"1b38",
"1b11",
"1b08",
"1b2d",
                                    0x1b17
                                    0x1b07
                                    0x1b37
                                    0x1b39
                                    0x1b13
                                    0x1b09
                                    0x1b33
                                    0x1b36
                                    0x1b38
                                    0x1b3a
                                    0x1b11
              "1b2d" ,
"1b20" ,
"1b15" ,
                                    0x1b2d
                                    0x1b20
                                    0x1b15
              "1b12" ,
                                    0x1b12 },
              "lb12", 0xlb12
"lb2e", 0xlb2e
"lb14", 0xlb14
"lb19", 0xlb19
"lb2f", 0xlb2f
"lb1e", 0xlb1e
"lb3e", 0xlb3e
"lb32", 0xlb32
"lb3c", 0xlb3c
                                    0x1b1e },
                                    0x1b3e },
```

Attention

when adding new hardware this file hat to be changed too.

In this structure array *models*[] for each device the name (the device id as string in hex without leading 0x) and its usb device id as short must be entered in this array.

Definition at line 622 of file usb linux.c.

```
5.39.4.3 struct udev* udev [static]
```

Definition at line 606 of file usb_linux.c.

```
5.39.4.4 pthread_t udevthread
```

Definition at line 609 of file usb_linux.c.

```
5.39.4.5 pthread_t usbthread
```

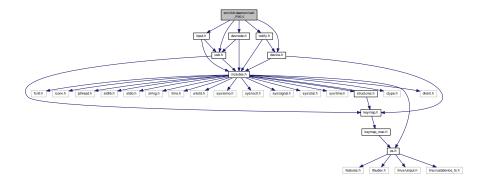
Todo These two thread vasriables seem to be unused: usbtread, udevthread

Definition at line 609 of file usb_linux.c.

5.40 src/ckb-daemon/usb mac.c File Reference

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

Include dependency graph for usb_mac.c:



Index

_model, 318	device.h, 30
_readlines_ctx, 48	ACT_M2
_DEFAULT_SOURCE	device.h, 30
os.h, 203	ACT_M3
_GNU_SOURCE	device.h, 30
os.h, 203	ACT_MR_RING
FILE_NOPATH	device.h, 30
includes.h, 87	ACT_NEXT
_cmd_get	device.h, 30
notify.c, 191	ACT_NEXT_NOWRAP
_cmd_macro	device.h, 30
input.c, 90	active
_freeprofile	devcmdunnamed, 8
profile.c, 205	allocprofile
ledthread	devcmdunnamed, 8
input_linux.c, 108	profile.c, 205
_mkdevpath	profile.h, 221
devnode.c, 49	,
_mknotifynode	BIND
devnode.c, 51	command.h, 20
nk95cmd	BR1
usb.h, 289	led_keyboard.c, 168
usb_linux.c, 319	BR2
resetusb	led_keyboard.c, 168
usb.c, 260	BR4
usb.h, 291	led_keyboard.c, 168
_rmnotifynode	BTN WHEELDOWN
devnode.c, 52	 keymap.h, <mark>121</mark>
_setupusb	BTN WHEELUP
usb.c, 260	keymap.h, 121
_start_dev	BUTTON_HID_COUNT
device.c, 26	keymap.c, 115
_updateconnected	bind
devnode.c, 52	devcmd. unnamed , 8
usbrecv	binding, 245
usb.c, 264	bit_reverse_table
usb.h, 291	led_keyboard.c, 175
usbsend	ica_keyboara.e, 170
usb.c. 266	CMD FIRST
•	command.h, 19
usb.h, 293	CMD LAST
ACCEL	command.h, 20
command.h, 19	CMD VT FIRST
ACTIVE	command.h, 19
command.h, 19	CLEAR KEYBIT
ACT LIGHT	structures.h, 254
device.h, 30	CMD COUNT
ACT LOCK	command.h, 18
-	CMD_DEV_COUNT
device.h, 30	
ACT_M1	command.h, 18

ckb_err	notify.c, 193
includes.h, 87	notify.h, 199
ckb_err_fn	cmd_hwload_kb
includes.h, 87	profile.h, 223
ckb_err_nofile	profile_keyboard.c, 235
includes.h, 87	cmd_hwload_mouse
ckb_fatal	profile mayo a 228
includes.h, 87 ckb_fatal_fn	profile_mouse.c, 238 cmd_hwsave_kb
includes.h, 87	profile.h, 225
ckb_fatal_nofile	profile_keyboard.c, 236
includes.h, 87	cmd_hwsave_mouse
ckb_info	profile.h, 226
includes.h, 87	profile_mouse.c, 239
ckb_info_fn	cmd_iauto
includes.h, 87	led.c, 129
ckb_info_nofile	led.h, 134
includes.h, 88	cmd_id
ckb_s_err	profile.c, 207
includes.h, 88	profile.h, 227
ckb_s_out	cmd_idle_kb
includes.h, 88	device.h, 32
ckb_warn	device_keyboard.c, 38
includes.h, 88	cmd_idle_mouse
ckb_warn_fn includes.h, 88	device.h, 32
ckb_warn_nofile	device_mouse.c, 42 cmd_inotify
includes.h, 88	led.c, 129
closeusb	led.h, 135
usb.c, 267	cmd_io_none
usb.h, 295	device_vtable.c, 45
cmd	cmd_ioff
command.h, 19	led.c, 129
cmd_active_kb	led.h, 135
device.h, 31	cmd_ion
device_keyboard.c, 38	led.c, 130
cmd_active_mouse	led.h, 136
device.h, 32	cmd_lift
device_mouse.c, 42	dpi.c, 69
cmd_bind	dpi.h, 74
input.c, 92	cmd_macro
input.h, 100	input.c, 92 input.h, 100
cmd_dpi dpi.c, 68	cmd_macro_none
dpi.h, 73	device_vtable.c, 45
cmd_dpisel	cmd name
dpi.c, 69	profile.c, 208
dpi.h, 74	profile.h, 227
cmd erase	cmd none
profile.c, 206	device_vtable.c, 45
profile.h, 222	cmd_notify
cmd_eraseprofile	notify.c, 194
profile.c, 207	notify.h, 199
profile.h, 223	cmd_pollrate
cmd_fwupdate	device.h, 33
firmware.c, 79	device_mouse.c, 43
firmware.h, 83	cmd_profileid
cmd_get	profile.c, 208

profile.h, 228	NOTIFYOFF, 19
cmd_profilename	NOTIFYON, 19
profile.c, 209	POLLRATE, 19
profile.h, 228	PROFILEID, 19
cmd_rebind	PROFILENAME, 19
input.c, 93	REBIND, 20
input.h, 100	RESTART, 20
cmd_restart	RGB, 19
notify.c, 194	SCROLLSPEED, 19
notify.h, 200	SNAP, 20
cmd rgb	SWITCH, 19
led.c, 130	UNBIND, 20
led.h, 136	command.c
cmd snap	cmd_strings, 16
dpi.c, 69	readcmd, 12
dpi.h, 74	TRY_WITH_RESET, 11
cmd_strings	command.h
command.c, 16	CMD COUNT, 18
cmd unbind	CMD_DEV_COUNT, 18
_	cmd, 19
input.c, 93	cmdhandler, 18
input.h, 101	cmdhandler_io, 19
cmdhandler	cmdhandler mac, 19
command.h, 18	devcmd, 19
cmdhandler_io	readcmd, 21
command.h, 19	
cmdhandler_mac	corsair_kbcopy
command.h, 19	keymap.c, 115
command.h	keymap.h, 124
ACCEL, 19	corsair_mousecopy
ACTIVE, 19	keymap.c, 116
BIND, 20	keymap.h, 124
CMD FIRST, 19	DELAY
CMD LAST, 20	
CMD VT FIRST, 19	command.h, 19 DITHER
DELAY, 19	
DITHER, 19	command.h, 19
DPI, 20	
DPISEL, 20	command.h, 20
ERASE, 19	DPISEL
ERASEPROFILE, 19	command.h, 20
FPS, 19	DELAY_LONG
FWUPDATE, 19	usb.h, 281
GET, 20	DELAY_MEDIUM
	usb.h, 281
HWLOAD, 19	DELAY_SHORT
HWSAVE, 19	usb.h, 281
IAUTO, 20	DEV_MAX
ID, 19	device.h, 30
IDLE, 19	DPI_COUNT
INOTIFY, 20	structures.h, 254
IOFF, 19	devcmd, 17
ION, 19	command.h, 19
LAYOUT, 19	devcmdunnamed, 7
LIFT, 20	active, 8
MACRO, 20	allocprofile, 8
MODE, 19	bind, 8
NAME, 19	dpi, <mark>8</mark>
NONE, 19	dpisel, 8
NOTIFY, 20	erase, 8
	•

eraseprofile, 8	imutex, 31
freeprofile, 8	inputmutex, 37
fwupdate, 8	keyboard, 37
get, 8	setactive, 31
hwload, 8	setactive_kb, 33
hwsave, 8	setactive_mouse, 35
iauto, 9	setmodeindex_nrgb, 36
id, 9	start_dev, 36
idle, 9	start_kb_nrgb, 37
inotify, 9	device_keyboard.c
ioff, 9	cmd_active_kb, 38
ion, 9	cmd_idle_kb, 38
lift, 9	setactive_kb, 39
loadprofile, 9	setmodeindex_nrgb, 41
macro, 9	start_kb_nrgb, 41
name, 9	device mouse.c
notify, 9	cmd_active_mouse, 42
pollrate, 9	cmd_idle_mouse, 42
profileid, 9	cmd pollrate, 43
profilename, 9	setactive_mouse, 43
rebind, 9	device_vtable.c
restart, 9	cmd io none, 45
rgb, 9	— — ·
_	cmd_macro_none, 45
setmodeindex, 9	cmd_none, 45
snap, 9	int1_int_none, 46
start, 9	int1_void_none, 46
unbind, 9	loadprofile_none, 46
updatedpi, 9	vtable_keyboard, 46
updateindicators, 9	vtable_keyboard_nonrgb, 46
updatergb, 9	vtable_mouse, 46
device.c	devlistmutex
_start_dev, 26	device.c, 28
devlistmutex, 28	devmain
devmutex, 28	usb.c, 269
hwload_mode, 28	devmutex
inputmutex, 28	device.c, 28
keyboard, 28	device.h, 37
start_dev, 27	devnode.c
device.h	_mkdevpath, 49
ACT_LIGHT, 30	_mknotifynode, 51
ACT_LOCK, 30	_rmnotifynode, 52
ACT_M1, 30	_updateconnected, 52
ACT_M2, 30	devpath, 60
ACT M3, 30	gid, 60
ACT MR RING, 30	MAX BUFFER, 48
ACT NEXT, 30	mkdevpath, 53
ACT NEXT NOWRAP, 30	mkfwnode, 54
cmd_active_kb, 31	mknotifynode, 55
cmd_active_mouse, 32	readlines, 55
cmd_idle_kb, 32	readlines_ctx_free, 56
cmd_idle_mouse, 32	readlines_ctx_init, 57
cmd_pollrate, 33	rm recursive, 57
DEV_MAX, 30	rmdevpath, 57
	·
devmutex, 37	rmnotifynode, 58
dmutex, 31	S_GID_READ, 49
IN_CORSAIR, 31	updateconnected, 59
IN_HID, 31	devnode.h
IS_CONNECTED, 31	devpath, 68

gid, 68	command.h, 19
mkdevpath, 62	FWUPDATE
mkfwnode, 62	command.h, 19
mknotifynode, 63	FEAT_ADJRATE
readlines, 64	structures.h, 254
readlines ctx, 62	FEAT ANSI
readlines ctx free, 65	structures.h, 254
readlines ctx init, 65	FEAT BIND
rmdevpath, 65	structures.h, 254
rmnotifynode, 66	FEAT_COMMON
S CUSTOM, 61	structures.h, 254
S_CUSTOM_R, 61	FEAT FWUPDATE
S READ, 61	structures.h, 254
S READDIR, 61	
S_READWRITE, 61	FEAT_FWVERSION
updateconnected, 67	structures.h, 255
devpath	FEAT_HWLOAD
devnode.c, 60	structures.h, 255
devnode.h, 68	FEAT_ISO
	structures.h, 255
dmutex	FEAT_LMASK
device.h, 31	structures.h, 255
dpi	FEAT_MONOCHROME
devcmdunnamed, 8	structures.h, 255
dpi.c	FEAT_MOUSEACCEL
cmd_dpi, 68	structures.h, 255
cmd_dpisel, 69	FEAT_NOTIFY
cmd_lift, 69	structures.h, 255
cmd_snap, 69	FEAT POLLRATE
loaddpi, 70	structures.h, 255
printdpi, 71	FEAT RGB
savedpi, 71	structures.h, 255
updatedpi, <mark>72</mark>	FEAT STD NRGB
dpi.h	structures.h, 255
cmd_dpi, 73	FEAT STD RGB
cmd_dpisel, 74	structures.h, 256
cmd_lift, 74	FW MAXSIZE
cmd_snap, 74	-
loaddpi, 75	firmware.c, 79
printdpi, 76	FW_NOFILE
savedpi, 76	firmware.c, 79
updatedpi, 77	FW_OK
dpisel	firmware.c, 79
devcmdunnamed, 8	FW_USBFAIL
dpiset, 246	firmware.c, 79
	FW_WRONGDEV
ERASE	firmware.c, 79
command.h, 19	features_mask
ERASEPROFILE	main.c, 189
command.h, 19	usb.c, 276
erase	firmware.c
devcmdunnamed, 8	cmd_fwupdate, 79
eraseprofile	FW_MAXSIZE, 79
devcmdunnamed, 8	FW_NOFILE, 79
euid_guard_start	FW_OK, 79
os.h, 203	FW USBFAIL, 79
euid_guard_stop	FW WRONGDEV, 79
os.h, 203	fwupdate, 80
03.11, 200	getfwversion, 82
FPS	firmware.h

cmd_fwupdate, 83	HWMODE_OR_RETURN
getfwversion, 84	notify.c, 190
freebind	has_key
input.c, 93	led.c, 131
input.h, 101	hid_kb_translate
freemode	keymap.c, 116
profile.c, 209	keymap.h, 124
freeprofile	hid_mouse_translate
devcmdunnamed, 8	keymap.c, 118
profile.c, 210	keymap.h, 126
profile.h, 229	hwload
fwupdate	devcmdunnamed, 8
devcmdunnamed, 8	hwload_mode
firmware.c, 80	device.c, 28
	main.c, 189
GET	usb.c, 276
command.h, 20	hwloadmode
get	profile_keyboard.c, 237
devcmdunnamed, 8	hwloadprofile
get_vtable	profile.h, 221
usb.c, 271	hwprofile, 250
getfwversion	hwsave
firmware.c, 82	devcmdunnamed, 8
firmware.h, 84	hwtonative
gethwmodename	profile.c, 214
profile.c, 210	profile.h, 233
profile.h, 229	promoni, 200
gethwprofilename	IAUTO
profile.c, 211	command.h, 20
profile.h, 230	ID
•	command.h, 19
getid	command.h, 19 IDLE
getid profile.c, 212	IDLE
getid profile.c, 212 profile.h, 231	
getid profile.c, 212 profile.h, 231 getmodename	IDLE command.h, 19 INOTIFY
getid profile.c, 212 profile.h, 231 getmodename profile.c, 212	IDLE command.h, 19
getid profile.c, 212 profile.h, 231 getmodename profile.c, 212 profile.h, 231	IDLE command.h, 19 INOTIFY command.h, 20 IOFF
getid profile.c, 212 profile.h, 231 getmodename profile.c, 212 profile.h, 231 getprofilename	IDLE command.h, 19 INOTIFY command.h, 20
getid profile.c, 212 profile.h, 231 getmodename profile.c, 212 profile.h, 231 getprofilename profile.c, 213	IDLE command.h, 19 INOTIFY command.h, 20 IOFF command.h, 19 ION
getid profile.c, 212 profile.h, 231 getmodename profile.c, 212 profile.h, 231 getprofilename profile.c, 213 profile.h, 232	IDLE command.h, 19 INOTIFY command.h, 20 IOFF command.h, 19
getid profile.c, 212 profile.h, 231 getmodename profile.c, 212 profile.h, 231 getprofilename profile.c, 213 profile.h, 232 gid	IDLE command.h, 19 INOTIFY command.h, 20 IOFF command.h, 19 ION command.h, 19 I_CAPS
getid profile.c, 212 profile.h, 231 getmodename profile.c, 212 profile.h, 231 getprofilename profile.c, 213 profile.h, 232 gid devnode.c, 60	IDLE command.h, 19 INOTIFY command.h, 20 IOFF command.h, 19 ION command.h, 19 I_CAPS structures.h, 256
getid profile.c, 212 profile.h, 231 getmodename profile.c, 212 profile.h, 231 getprofilename profile.c, 213 profile.h, 232 gid	IDLE command.h, 19 INOTIFY command.h, 20 IOFF command.h, 19 ION command.h, 19 I_CAPS structures.h, 256 I_NUM
getid profile.c, 212 profile.h, 231 getmodename profile.c, 212 profile.h, 231 getprofilename profile.c, 213 profile.h, 232 gid devnode.c, 60	IDLE command.h, 19 INOTIFY command.h, 20 IOFF command.h, 19 ION command.h, 19 I_CAPS structures.h, 256 I_NUM structures.h, 256
getid profile.c, 212 profile.h, 231 getmodename profile.c, 212 profile.h, 231 getprofilename profile.c, 213 profile.h, 232 gid devnode.c, 60 devnode.h, 68 HWLOAD	IDLE command.h, 19 INOTIFY command.h, 20 IOFF command.h, 19 ION command.h, 19 I_CAPS structures.h, 256 I_NUM structures.h, 256 I_SCROLL
getid profile.c, 212 profile.h, 231 getmodename profile.c, 212 profile.h, 231 getprofilename profile.c, 213 profile.h, 232 gid devnode.c, 60 devnode.h, 68 HWLOAD command.h, 19	IDLE command.h, 19 INOTIFY command.h, 20 IOFF command.h, 19 ION command.h, 19 I_CAPS structures.h, 256 I_NUM structures.h, 256 I_SCROLL structures.h, 256
getid profile.c, 212 profile.h, 231 getmodename profile.c, 212 profile.h, 231 getprofilename profile.c, 213 profile.h, 232 gid devnode.c, 60 devnode.h, 68 HWLOAD command.h, 19 HWSAVE	IDLE command.h, 19 INOTIFY command.h, 20 IOFF command.h, 19 ION command.h, 19 I_CAPS structures.h, 256 I_NUM structures.h, 256 I_SCROLL structures.h, 256 IFACE_MAX
getid profile.c, 212 profile.h, 231 getmodename profile.c, 212 profile.h, 231 getprofilename profile.c, 213 profile.h, 232 gid devnode.c, 60 devnode.h, 68 HWLOAD command.h, 19 HWSAVE command.h, 19	IDLE command.h, 19 INOTIFY command.h, 20 IOFF command.h, 19 ION command.h, 19 I_CAPS structures.h, 256 I_NUM structures.h, 256 I_SCROLL structures.h, 256 IFACE_MAX structures.h, 256
getid profile.c, 212 profile.h, 231 getmodename profile.c, 212 profile.h, 231 getprofilename profile.c, 213 profile.h, 232 gid devnode.c, 60 devnode.h, 68 HWLOAD command.h, 19 HWSAVE command.h, 19 HAS_ANY_FEATURE	IDLE command.h, 19 INOTIFY command.h, 20 IOFF command.h, 19 ION command.h, 19 I_CAPS structures.h, 256 I_NUM structures.h, 256 I_SCROLL structures.h, 256 IFACE_MAX structures.h, 256 IN_CORSAIR
getid profile.c, 212 profile.h, 231 getmodename profile.c, 212 profile.h, 231 getprofilename profile.c, 213 profile.h, 232 gid devnode.c, 60 devnode.h, 68 HWLOAD command.h, 19 HWSAVE command.h, 19 HAS_ANY_FEATURE structures.h, 256	IDLE command.h, 19 INOTIFY command.h, 20 IOFF command.h, 19 ION command.h, 19 I_CAPS structures.h, 256 I_NUM structures.h, 256 I_SCROLL structures.h, 256 IFACE_MAX structures.h, 256 IN_CORSAIR device.h, 31
getid profile.c, 212 profile.h, 231 getmodename profile.c, 212 profile.h, 231 getprofilename profile.c, 213 profile.h, 232 gid devnode.c, 60 devnode.h, 68 HWLOAD command.h, 19 HWSAVE command.h, 19 HAS_ANY_FEATURE structures.h, 256 HAS_FEATURES	IDLE command.h, 19 INOTIFY command.h, 20 IOFF command.h, 19 ION command.h, 19 I_CAPS structures.h, 256 I_NUM structures.h, 256 I_SCROLL structures.h, 256 IFACE_MAX structures.h, 256 IN_CORSAIR device.h, 31 IN_HID
getid profile.c, 212 profile.h, 231 getmodename profile.c, 212 profile.h, 231 getprofilename profile.c, 213 profile.h, 232 gid devnode.c, 60 devnode.h, 68 HWLOAD command.h, 19 HWSAVE command.h, 19 HAS_ANY_FEATURE structures.h, 256 HAS_FEATURES structures.h, 256	IDLE command.h, 19 INOTIFY command.h, 20 IOFF command.h, 19 ION command.h, 19 I_CAPS structures.h, 256 I_NUM structures.h, 256 I_SCROLL structures.h, 256 IFACE_MAX structures.h, 256 IN_CORSAIR device.h, 31 IN_HID device.h, 31
getid profile.c, 212 profile.h, 231 getmodename profile.c, 212 profile.h, 231 getprofilename profile.c, 213 profile.h, 232 gid devnode.c, 60 devnode.h, 68 HWLOAD command.h, 19 HWSAVE command.h, 19 HAS_ANY_FEATURE structures.h, 256 HAS_FEATURES structures.h, 256 HW_STANDARD	IDLE command.h, 19 INOTIFY command.h, 20 IOFF command.h, 19 ION command.h, 19 I_CAPS structures.h, 256 I_NUM structures.h, 256 I_SCROLL structures.h, 256 IFACE_MAX structures.h, 256 IN_CORSAIR device.h, 31 IN_HID device.h, 31 INDEX_OF
getid profile.c, 212 profile.h, 231 getmodename profile.c, 212 profile.h, 231 getprofilename profile.c, 213 profile.h, 232 gid devnode.c, 60 devnode.h, 68 HWLOAD command.h, 19 HWSAVE command.h, 19 HAS_ANY_FEATURE structures.h, 256 HAS_FEATURES structures.h, 256 HW_STANDARD notify.c, 190	IDLE command.h, 19 INOTIFY command.h, 20 IOFF command.h, 19 ION command.h, 19 I_CAPS structures.h, 256 I_NUM structures.h, 256 I_SCROLL structures.h, 256 IFACE_MAX structures.h, 256 IN_CORSAIR device.h, 31 IN_HID device.h, 31 INDEX_OF includes.h, 88
getid profile.c, 212 profile.h, 231 getmodename profile.c, 212 profile.h, 231 getprofilename profile.c, 213 profile.h, 232 gid devnode.c, 60 devnode.h, 68 HWLOAD command.h, 19 HWSAVE command.h, 19 HAS_ANY_FEATURE structures.h, 256 HAS_FEATURES structures.h, 256 HW_STANDARD notify.c, 190 HWMODE_K70	IDLE command.h, 19 INOTIFY command.h, 20 IOFF command.h, 19 ION command.h, 19 I_CAPS structures.h, 256 I_NUM structures.h, 256 I_SCROLL structures.h, 256 I_FACE_MAX structures.h, 256 IN_CORSAIR device.h, 31 IN_HID device.h, 31 INDEX_OF includes.h, 88 IS_CONNECTED
getid profile.c, 212 profile.h, 231 getmodename profile.c, 212 profile.h, 231 getprofilename profile.c, 213 profile.h, 232 gid devnode.c, 60 devnode.h, 68 HWLOAD command.h, 19 HWSAVE command.h, 19 HAS_ANY_FEATURE structures.h, 256 HAS_FEATURES structures.h, 256 HW_STANDARD notify.c, 190 HWMODE_K70 structures.h, 256	IDLE command.h, 19 INOTIFY command.h, 20 IOFF command.h, 19 ION command.h, 19 I_CAPS structures.h, 256 I_NUM structures.h, 256 I_SCROLL structures.h, 256 I_SCROLL structures.h, 256 IFACE_MAX structures.h, 256 IN_CORSAIR device.h, 31 IN_HID device.h, 31 INDEX_OF includes.h, 88 IS_CONNECTED device.h, 31
getid profile.c, 212 profile.h, 231 getmodename profile.c, 212 profile.h, 231 getprofilename profile.c, 213 profile.h, 232 gid devnode.c, 60 devnode.h, 68 HWLOAD command.h, 19 HWSAVE command.h, 19 HAS_ANY_FEATURE structures.h, 256 HAS_FEATURES structures.h, 256 HW_STANDARD notify.c, 190 HWMODE_K70 structures.h, 256 HWMODE_K95	IDLE command.h, 19 INOTIFY command.h, 20 IOFF command.h, 19 ION command.h, 19 I_CAPS structures.h, 256 I_NUM structures.h, 256 I_SCROLL structures.h, 256 I_SCROLL structures.h, 256 IFACE_MAX structures.h, 256 IN_CORSAIR device.h, 31 IN_HID device.h, 31 INDEX_OF includes.h, 88 IS_CONNECTED device.h, 31 IS_FULLRANGE
getid profile.c, 212 profile.h, 231 getmodename profile.c, 212 profile.h, 231 getprofilename profile.c, 213 profile.h, 232 gid devnode.c, 60 devnode.h, 68 HWLOAD command.h, 19 HWSAVE command.h, 19 HAS_ANY_FEATURE structures.h, 256 HAS_FEATURES structures.h, 256 HW_STANDARD notify.c, 190 HWMODE_K70 structures.h, 256 HWMODE_K95 structures.h, 256	IDLE command.h, 19 INOTIFY command.h, 20 IOFF command.h, 19 ION command.h, 19 I_CAPS structures.h, 256 I_NUM structures.h, 256 I_SCROLL structures.h, 256 IFACE_MAX structures.h, 256 IN_CORSAIR device.h, 31 IN_HID device.h, 31 INDEX_OF includes.h, 88 IS_CONNECTED device.h, 31 IS_FULLRANGE usb.h, 281
getid profile.c, 212 profile.h, 231 getmodename profile.c, 212 profile.h, 231 getprofilename profile.c, 213 profile.h, 232 gid devnode.c, 60 devnode.h, 68 HWLOAD command.h, 19 HWSAVE command.h, 19 HAS_ANY_FEATURE structures.h, 256 HAS_FEATURES structures.h, 256 HW_STANDARD notify.c, 190 HWMODE_K70 structures.h, 256 HWMODE_K95	IDLE command.h, 19 INOTIFY command.h, 20 IOFF command.h, 19 ION command.h, 19 I_CAPS structures.h, 256 I_NUM structures.h, 256 I_SCROLL structures.h, 256 I_SCROLL structures.h, 256 IFACE_MAX structures.h, 256 IN_CORSAIR device.h, 31 IN_HID device.h, 31 INDEX_OF includes.h, 88 IS_CONNECTED device.h, 31 IS_FULLRANGE

IS_K70	timespec_lt, 89
usb.h, 282	uchar, 89
IS_K95	ushort, 89
usb.h, 282	initbind
IS_M65	input.c, 94
usb.h, 282	input.h, 101
IS_MOD	initmode
input.h, 99	profile.c, 214
IS MONOCHROME	inotify
usb.h, 282	devcmdunnamed, 9
IS MONOCHROME DEV	input.c
usb.h, 282	_cmd_macro, 90
IS MOUSE	cmd_bind, 92
usb.h, 282	cmd macro, 92
IS MOUSE DEV	cmd_rebind, 93
usb.h, 282	cmd_unbind, 93
IS RGB	freebind, 93
usb.h, 282	IS WHEEL, 90
IS RGB DEV	initbind, 94
usb.h, 283	inputupdate, 94
IS SABRE	
-	inputupdate_keys, 95
usb.h, 283	macromask, 97
IS_SCIMITAR	updateindicators_kb, 98
usb.h, 283	input.h
IS_STRAFE	cmd_bind, 100
usb.h, 283	cmd_macro, 100
IS_WHEEL	cmd_rebind, 100
input.c, 90	cmd_unbind, 101
iauto	freebind, 101
devcmdunnamed, 9	IS_MOD, 99
id	initbind, 101
devcmdunnamed, 9	inputupdate, 102
idle	os_inputclose, 103
devcmdunnamed, 9	os_inputopen, 103
imutex	os_keypress, 104
device.h, 31	os_mousemove, 105
includes.h	os_setupindicators, 106
FILE_NOPATH, 87	updateindicators_kb, 107
ckb_err, 87	input_linux.c
ckb_err_fn, 87	_ledthread, 108
ckb_err_nofile, 87	isync, 109
ckb_fatal, 87	os_inputclose, 109
ckb_fatal_fn, 87	os_inputopen, 110
ckb_fatal_nofile, 87	os_keypress, 111
ckb_info, 87	os_mousemove, 111
ckb info fn, 87	os_setupindicators, 112
ckb_info_nofile, 88	uinputopen, 113
ckb_s_err, 88	inputmutex
ckb_s_out, 88	device.c, 28
ckb_warn, 88	device.h, 37
ckb_warn_fn, 88	inputupdate
ckb_warn_nofile, 88	input.c, 94
INDEX_OF, 88	input.h, 102
timespec_add, 89	inputupdate_keys
timespec_eq, 88	input.c, 95
timespec_eq, 88	int1_int_none
timespec_ge, 89	device_vtable.c, 46
timespec_le, 89	int1_void_none
umespec_16, 00	III.I_void_Hone

device_vtable.c, 46	N_KEYBYTES_HW, 122
ioff	N_KEYBYTES_INPUT, 123
devcmdunnamed, 9	N_KEYS_EXTENDED, 123
ion	N_KEYS_EXTRA, 123
devcmdunnamed, 9	N_KEYS_HW, 123
isblack	N_KEYS_INPUT, 123
led_mouse.c, 177	N_MOUSE_ZONES, 123
iselect	SCAN_KBD, 123
led.c, 131	SCAN_MOUSE, 123
isync	SCAN_SILENT, 123
input_linux.c, 109	LAYOUT
KB NAME LEN	command.h, 19
structures.h, 257	LIFT
KEY BACKSLASH ISO	command.h, 20
keymap.h, 121	LED_DPI
KEY_CORSAIR	keymap.h, 122
keymap.h, 121	LED_MOUSE
KEY_NONE	keymap.h, 122
keymap.h, 121	LIFT_MAX
KEY_UNBOUND	structures.h, 257
keymap.h, 122	LIFT_MIN
kbsyspath	structures.h, 257
usb_linux.c, 339	led.c
key, 120	cmd_iauto, 129
keyboard	cmd_inotify, 129
device.c, 28	cmd_ioff, 129
device.h, 37	cmd_ion, 130
keymacro, 244	cmd_rgb, 130
keymap	has_key, 131
keymap.c, 119	iselect, 131
keymap.h, 127	printrgb, 132 led.h
keymap.c	
BUTTON_HID_COUNT, 115 corsair_kbcopy, 115	cmd_iauto, 134 cmd_inotify, 135
corsair_nousecopy, 116	cmd_inothy, 135
hid kb translate, 116	cmd_ion, 136
hid_mouse_translate, 118	cmd_rgb, 136
keymap, 119	loadrgb_kb, 136
keymap.h	loadrgb_mouse, 138
BTN_WHEELDOWN, 121	printrgb, 139
BTN_WHEELUP, 121	savergb_kb, 140
corsair_kbcopy, 124	savergb_mouse, 142
corsair_mousecopy, 124	updatergb_kb, 142
hid_kb_translate, 124	updatergb_mouse, 144
hid_mouse_translate, 126	led_keyboard.c
KEY_BACKSLASH_ISO, 121	BR1, 168
KEY_CORSAIR, 121	BR2, 168
KEY_NONE, 121	BR4, 168
KEY_UNBOUND, 122	bit_reverse_table, 175
keymap, 127	loadrgb_kb, 169
LED_DPI, 122	makergb_512, 170
LED_MOUSE, 122	makergb_full, 171
MOUSE_BUTTON_FIRST, 122	O0, 168
MOUSE_EXTRA_FIRST, 122	O1, 168
N_BUTTONS_EXTENDED, 122	O2, 168
N_BUTTONS_HW, 122 N_KEY_ZONES, 122	O3, 168 O4, 168
N_KEY_ZONES, 122 N_KEYBYTES_EXTENDED, 122	O5, 168
N_NEIDTTES_EATENDED, 122	03, 100

00.400	
O6, 168	main.c, 182
07, 168	main.c
O8, 168	features_mask, 189
ordered8to3, 172	hwload_mode, 189
quantize8to3, 172	localecase, 181
rgbcmp, 172	main, 1 <mark>82</mark>
savergb_kb, 173	main_ac, 189
updatergb_kb, 174	main_av, 189
led_mouse.c	quit, 184
isblack, 177	quitWithLock, 185
loadrgb_mouse, 178	reset stop, 189
rgbcmp, 178	restart, 186
savergb_mouse, 179	sighandler, 187
updatergb_mouse, 179	sighandler2, 188
lift	timespec_add, 188
devcmdunnamed, 9	main_ac
lighting, 247	main.c, 189
	,
loaddpi	main_av
dpi.c, 70	main.c, 189
dpi.h, 75	makergb_512
loadprofile	led_keyboard.c, 170
devcmdunnamed, 9	makergb_full
profile.c, 215	led_keyboard.c, 171
profile.h, 233	mkdevpath
loadprofile_none	devnode.c, 53
device_vtable.c, 46	devnode.h, 62
loadrgb_kb	mkfwnode
led.h, 136	devnode.c, 54
led_keyboard.c, 169	devnode.h, 62
loadrgb_mouse	mknotifynode
led.h, 138	devnode.c, 55
led_mouse.c, 178	devnode.h, 63
localecase	models
main.c, 181	usb_linux.c, 339
mam.c, 101	usb_iiiux.c, 300
MACRO	NAME
command.h, 20	command.h, 19
MODE	NONE
command.h, 19	command.h, 19
MACRO MAX	NOTIFY
structures.h, 257	command.h, 20
MAX BUFFER	NOTIFYOFF
_	
devnode.c, 48	command.h, 19 NOTIFYON
MD_NAME_LEN	
structures.h, 257	command.h, 19
MODE_COUNT	N_BUTTONS_EXTENDED
structures.h, 257	keymap.h, 122
MOUSE_BUTTON_FIRST	N_BUTTONS_HW
keymap.h, 122	keymap.h, <mark>122</mark>
MOUSE_EXTRA_FIRST	N_KEY_ZONES
keymap.h, 122	keymap.h, 122
MSG_SIZE	N_KEYBYTES_EXTENDED
structures.h, 257	keymap.h, 122
macro	N_KEYBYTES_HW
devcmdunnamed, 9	keymap.h, 122
macroaction, 243	N_KEYBYTES_INPUT
macromask	keymap.h, 123
input.c, 97	N KEYS EXTENDED
main	keymap.h, 123
	,, , , , , , , , , , , , , , ,

N_KEYS_EXTRA	00
keymap.h, 123	led_keyboard.c, 168
N_KEYS_HW	01
keymap.h, 123	led_keyboard.c, 168
N_KEYS_INPUT	02
keymap.h, 123 N MODELS	led_keyboard.c, 168 O3
usb linux.c, 318	led_keyboard.c, 168
N MOUSE ZONES	O4
keymap.h, 123	led_keyboard.c, 168
NEEDS_FW_UPDATE	O5
structures.h, 257	led_keyboard.c, 168
NK95_HWOFF	O6
usb.h, 283	led_keyboard.c, 168
NK95_HWON	07
usb.h, 283	led_keyboard.c, 168
NK95_M1	O8
usb.h, 283	led_keyboard.c, 168
NK95_M2	OUTFIFO_MAX
usb.h, 283	structures.h, 257
NK95_M3	ordered8to3
usb.h, 283	led_keyboard.c, 172
name	os.h
devcmdunnamed, 9	_DEFAULT_SOURCE, 203
nativetohw	_GNU_SOURCE, 203
profile.c, 215	euid_guard_start, 203
profile.h, 234	euid_guard_stop, 203
nk95cmd	UINPUT_VERSION, 203
usb.h, 284	os_closeusb
notify	usb.h, 296
devcmdunnamed, 9	usb_linux.c, 320
notify.c	os_inputclose
_cmd_get, 191	input.h, 103
cmd_get, 193	input_linux.c, 109
cmd_notify, 194	os_inputmain
cmd_restart, 194	usb.h, 297
HW_STANDARD, 190	usb_linux.c, 321
HWMODE_OR_RETURN, 190	os_inputopen
nprintf, 195	input.h, 103
nprintind, 195	input_linux.c, 110
nprintkey, 196	os_keypress
restart, 197	input.h, 104
notify.h	input_linux.c, 111
cmd_get, 199	os_mousemove
cmd_notify, 199	input.h, 105
cmd_restart, 200	input_linux.c, 111
nprintf, 200	os_resetusb
nprintind, 201	usb.h, 301
nprintkey, 202	usb_linux.c, 324
nprintf	os_sendindicators
notify.c, 195	usb.h, 302
notify.h, 200	usb_linux.c, 325
nprintind	os_setupindicators
notify.c, 195	input.h, 106
notify.h, 201	
nprintkey	input_linux.c, 112
· · · · · ·	os_setupusb
notify.c, 196 notify.h, 202	• —

os_usbrecv	P_K95_NRGB
usb.h, 304	usb.h, 286
usb_linux.c, 327	P_K95_NRGB_STR
os_usbsend	usb.h, 286
usb.h, 305	P_K95_PLATINUM
usb_linux.c, 328	usb.h, 286
DOLL DATE	P_K95_PLATINUM_STR
POLLRATE	usb.h, 286
command.h, 19	P_K95_STR
PROFILEID	usb.h, 286
command.h, 19 PROFILENAME	P_M65
command.h, 19	usb.h, 286
P_HARPOON	P_M65_PRO
usb.h, 284	usb.h, 286
P_HARPOON_STR	P_M65_PRO_STR
usb.h, 284	usb.h, 287
P K65	P_M65_STR
usb.h, 284	usb.h, 287
P_K65_LUX	P_SABRE_L
usb.h, 284	usb.h, 287
P K65 LUX STR	P_SABRE_L_STR
usb.h, 284	usb.h, 287
P_K65_NRGB	P_SABRE_N
usb.h, 284	usb.h, 287
P_K65_NRGB_STR	P_SABRE_N_STR
usb.h, 284	usb.h, 287
P_K65_RFIRE	P_SABRE_O
usb.h, 284	usb.h, 287
P_K65_RFIRE_STR	P_SABRE_O2
usb.h, 284	usb.h, 287
P_K65_STR	P_SABRE_O2_STR usb.h, 287
usb.h, 285	P_SABRE_O_STR
P_K70	usb.h, 287
usb.h, 285	P SCIMITAR
P_K70_LUX	usb.h, 287
usb.h, 285	P_SCIMITAR_PRO
P_K70_LUX_NRGB	usb.h, 288
usb.h, 285	P_SCIMITAR_PRO_STR
P_K70_LUX_NRGB_STR	usb.h, 288
usb.h, 285 P_K70_LUX_STR	P_SCIMITAR_STR
usb.h, 285	usb.h, 288
P K70 NRGB	P STRAFE
usb.h, 285	usb.h, 288
P_K70_NRGB_STR	P_STRAFE_NRGB
usb.h, 285	usb.h, 288
P K70 RFIRE	P_STRAFE_NRGB_STR
usb.h, 285	usb.h, 288
P K70 RFIRE NRGB	P_STRAFE_STR
usb.h, 285	usb.h, 288
P_K70_RFIRE_NRGB_STR	PR_NAME_LEN
usb.h, 286	structures.h, 258
P_K70_RFIRE_STR	pollrate
usb.h, 286	devcmdunnamed, 9
P_K70_STR	printdpi
usb.h, 286	dpi.c, 71
P_K95	dpi.h, 76
usb.h, 286	printname

profile.c, 216	profile_keyboard.c
printrgb	cmd_hwload_kb, 235
led.c, 132	cmd_hwsave_kb, 236
led.h, 139	hwloadmode, 237
product_str	profile_mouse.c
usb.c, 272	cmd_hwload_mouse, 238
usb.h, 308	cmd_hwsave_mouse, 239
profile.c	profileid
_freeprofile, 205	devcmdunnamed, 9
allocprofile, 205	profilename
cmd_erase, 206	devcmdunnamed, 9
cmd_eraseprofile, 207	
cmd_id, 207	quantize8to3
cmd_name, 208	led_keyboard.c, 172
cmd_profileid, 208	quit
cmd_profilename, 209	main.c, 184
freemode, 209	quitWithLock
freeprofile, 210	main.c, 185
gethwmodename, 210	DEDIND
gethwprofilename, 211	REBIND
getid, 212	command.h, 20
getmodename, 212	RESTART
getprofilename, 213	command.h, 20
hwtonative, 214	RGB
initmode, 214	command.h, 19
loadprofile, 215	readcmd
nativetohw, 215	command.c, 12
printname, 216	command.h, 21
setid, 217	readlines
u16dec, 217	devnode.c, 55
u16enc, 218	devnode.h, 64
urldecode2, 218	readlines_ctx
urlencode2, 219	devnode.h, 62
	readlines_ctx_free
utf16to8, 220	devnode.c, 56
utf8to16, 220 profile.h	devnode.h, 65
•	readlines_ctx_init
allocprofile, 221	devnode.c, 57
cmd_erase, 222	devnode.h, 65
cmd_eraseprofile, 223	rebind
cmd_hwload_kb, 223	devcmdunnamed, 9
cmd_hwload_mouse, 224	reset_stop
cmd_hwsave_kb, 225	main.c, 189
cmd_hwsave_mouse, 226	usb.c, 276
cmd_id, 227	resetusb
cmd_name, 227	usb.h, 288
cmd_profileid, 228	restart
cmd_profilename, 228	devcmdunnamed, 9
freeprofile, 229	main.c, 186
gethwmodename, 229	notify.c, 197
gethwprofilename, 230	revertusb
getid, 231	usb.c, 273
getmodename, 231	usb.h, 309
getprofilename, 232	rgb
hwloadprofile, 221	devcmdunnamed, 9
hwtonative, 233	rgbcmp
loadprofile, 233	led_keyboard.c, 172
nativetohw, 234	led_mouse.c, 178
setid, 234	rm_recursive

devnode.c, 57	setid
rmdevpath	profile.c, 217
devnode.c, 57	profile.h, 234
devnode.h, 65	setmodeindex
rmnotifynode	devcmdunnamed, 9
devnode.c, 58	setmodeindex_nrgb
devnode.h, 66	device.h, 36
SCROLLSPEED	device_keyboard.c, 41
command.h, 19	setupusb
SNAP	usb.c, 274
command.h, 20	usb.h, 311
SWITCH	sighandler
command.h, 19	main.c, 187
S CUSTOM	sighandler2
devnode.h, 61	main.c, 188
S CUSTOM R	snap
	devcmdunnamed, 9
devnode.h, 61	src/ckb-daemon/command.c, 11
S_GID_READ	src/ckb-daemon/command.h, 16
devnode.c, 49	src/ckb-daemon/device.c, 25
S_READ	src/ckb-daemon/device.h, 28
devnode.h, 61	src/ckb-daemon/device_keyboard.c, 38
S_READDIR	src/ckb-daemon/device_mouse.c, 41
devnode.h, 61	src/ckb-daemon/device_vtable.c, 44
S_READWRITE	src/ckb-daemon/devnode.c, 47
devnode.h, 61	src/ckb-daemon/devnode.h, 60
SCAN_KBD	src/ckb-daemon/dpi.c, 68
keymap.h, 123	src/ckb-daemon/dpi.h, 73
SCAN_MOUSE	src/ckb-daemon/extra_mac.c, 78
keymap.h, 123	src/ckb-daemon/firmware.c, 78
SCAN_SILENT	src/ckb-daemon/firmware.h, 83
keymap.h, 123	src/ckb-daemon/includes.h, 85
SCROLL_ACCELERATED	src/ckb-daemon/input.c, 89
structures.h, 258	src/ckb-daemon/input.h, 98
SCROLL_MAX	src/ckb-daemon/input_linux.c, 107
structures.h, 258	src/ckb-daemon/input_mac.c, 107
SCROLL_MIN	• —
structures.h, 258	src/ckb-daemon/input_mac_mouse.c, 114
SERIAL_LEN	src/ckb-daemon/keymap.c, 114
structures.h, 258	src/ckb-daemon/keymap.h, 119
SET_KEYBIT	src/ckb-daemon/keymap_mac.h, 127
structures.h, 258	src/ckb-daemon/led.c, 128
savedpi	src/ckb-daemon/led.h, 134
dpi.c, 71	src/ckb-daemon/led_keyboard.c, 145
dpi.h, 76	src/ckb-daemon/led_mouse.c, 177
savergb_kb	src/ckb-daemon/main.c, 180
led.h, 140	src/ckb-daemon/notify.c, 189
led_keyboard.c, 173	src/ckb-daemon/notify.h, 198
savergb_mouse	src/ckb-daemon/os.h, 202
led.h, 142	src/ckb-daemon/profile.c, 204
led_mouse.c, 179	src/ckb-daemon/profile.h, 220
setactive	src/ckb-daemon/profile_keyboard.c, 235
device.h, 31	src/ckb-daemon/profile_mouse.c, 238
setactive_kb	src/ckb-daemon/structures.h, 240
device.h, 33	src/ckb-daemon/usb.c, 259
device_keyboard.c, 39	src/ckb-daemon/usb.h, 277
setactive_mouse	src/ckb-daemon/usb_linux.c, 316
device.h, 35	src/ckb-daemon/usb_mac.c, 340
device_mouse.c, 43	start

devcmdunnamed, 9	command.c, 11
start_dev	timespec_add
device.c, 27	includes.h, 89
device.h, 36	main.c, 188
start_kb_nrgb	timespec_eq
device.h, 37	includes.h, 88
device_keyboard.c, 41	timespec_ge
strtrim	includes.h, 88
usb_linux.c, 330	timespec_gt
structures.h	includes.h, 89
CLEAR KEYBIT, 254	timespec_le
DPI COUNT, 254	includes.h, 89
FEAT ADJRATE, 254	timespec_lt
FEAT ANSI, 254	includes.h, 89
FEAT BIND, 254	,
FEAT COMMON, 254	u16dec
FEAT FWUPDATE, 254	profile.c, 217
FEAT FWVERSION, 255	u16enc
-	profile.c, 218
FEAT_HWLOAD, 255	UNBIND
FEAT_ISO, 255	command.h, 20
FEAT_LMASK, 255	UINPUT_VERSION
FEAT_MONOCHROME, 255	
FEAT_MOUSEACCEL, 255	os.h, 203
FEAT_NOTIFY, 255	USB_DELAY_DEFAULT
FEAT_POLLRATE, 255	usb.h, 288
FEAT_RGB, 255	uchar
FEAT_STD_NRGB, 255	includes.h, 89
FEAT_STD_RGB, 256	udev
HAS_ANY_FEATURE, 256	usb_linux.c, 340
HAS FEATURES, 256	udev_enum
HWMODE_K70, 256	usb_linux.c, 331
HWMODE_K95, 256	udevthread
HWMODE_MAX, 256	usb_linux.c, 340
I_CAPS, 256	uinputopen
I_O/11 6, 200	input linux.c, 113
I_SCROLL, 256	unbind
IFACE_MAX, 256	devcmdunnamed, 9
	updateconnected
KB_NAME_LEN, 257	devnode.c, 59
LIFT_MAX, 257	devnode.h, 67
LIFT_MIN, 257	updatedpi
MACRO_MAX, 257	devcmdunnamed, 9
MD_NAME_LEN, 257	dpi.c, 72
MODE_COUNT, 257	
MSG_SIZE, 257	dpi.h, 77
NEEDS_FW_UPDATE, 257	updateindicators
OUTFIFO_MAX, 257	devcmdunnamed, 9
PR_NAME_LEN, 258	updateindicators_kb
SCROLL_ACCELERATED, 258	input.c, 98
SCROLL_MAX, 258	input.h, 107
SCROLL_MIN, 258	updatergb
SERIAL LEN, 258	devcmdunnamed, 9
SET KEYBIT, 258	updatergb_kb
vtable_keyboard, 258	led.h, 142
vtable_keyboard_nonrgb, 258	led_keyboard.c, 174
vtable_neyboard_normgb, 256	updatergb_mouse
vidule_iniouse, 200	led.h, 144
TEST RESET	led_mouse.c, 179
usb_linux.c, 318	urldecode2
TRY WITH RESET	profile.c, 218
	promo.o, = 10

	D 1/25 110/ 55/
urlencode2	P_K65_LUX, 284
profile.c, 219	P_K65_LUX_STR, 284
usb.c	P_K65_NRGB, 284
_resetusb, 260	P_K65_NRGB_STR, 284
_setupusb, 260	P_K65_RFIRE, 284
_usbrecv, 264	P_K65_RFIRE_STR, 284
_usbsend, 266	P_K65_STR, 285
closeusb, 267	P_K70, 285
devmain, 269	P_K70_LUX, 285
features_mask, 276	P_K70_LUX_NRGB, 285
get_vtable, 271	P_K70_LUX_NRGB_STR, 285
hwload_mode, 276	P_K70_LUX_STR, 285
product_str, 272	P_K70_NRGB, 285
reset_stop, 276	P_K70_NRGB_STR, 285
revertusb, 273	P K70 RFIRE, 285
setupusb, 274	P_K70_RFIRE_NRGB, 285
usb_tryreset, 275	P K70 RFIRE STR, 286
usbmutex, 277	P K70 STR, 286
vendor_str, 276	P K95, 286
usb.h	P K95 NRGB, 286
_nk95cmd, 289	P K95 NRGB STR, 286
_resetusb, 291	P_K95_PLATINUM, 286
_usbrecv, 291	P_K95_PLATINUM_STR, 286
_usbsend, 293	P_K95_STR, 286
closeusb, 295	P_M65, 286
DELAY_LONG, 281	P_M65_PRO, 286
DELAY_MEDIUM, 281	P_M65_PRO_STR, 287
DELAY_SHORT, 281	P_M65_STR, 287
IS_FULLRANGE, 281	P_SABRE_L, 287
IS_K65, 281	P_SABRE_L_STR, 287
IS_K70, 282	P_SABRE_N, 287
IS_K95, 282	P_SABRE_N_STR, 287
IS_M65, 282	P_SABRE_O, 287
IS_MONOCHROME, 282	P_SABRE_O2, 287
IS_MONOCHROME_DEV, 282	P_SABRE_O2_STR, 287
IS_MOUSE, 282	P_SABRE_O_STR, 287
IS_MOUSE_DEV, 282	P_SCIMITAR, 287
IS_RGB, 282	P_SCIMITAR_PRO, 288
IS_RGB_DEV, 283	P_SCIMITAR_PRO_STR, 288
IS_SABRE, 283	P SCIMITAR STR, 288
IS SCIMITAR, 283	P STRAFE, 288
IS_STRAFE, 283	P STRAFE NRGB, 288
NK95_HWOFF, 283	P_STRAFE_NRGB_STR, 288
NK95 HWON, 283	P STRAFE STR, 288
NK95_M1, 283	product_str, 308
NK95_M2, 283	resetusb, 288
NK95_M3, 283	
<i>= '</i>	revertusb, 309
nk95cmd, 284	setupusb, 311
os_closeusb, 296	USB_DELAY_DEFAULT, 288
os_inputmain, 297	usb_tryreset, 312
os_resetusb, 301	usbkill, 314
os_sendindicators, 302	usbmain, 314
os_setupusb, 302	usbrecv, 288
os_usbrecv, 304	usbsend, 289
os_usbsend, 305	V_CORSAIR, 289
P_HARPOON, 284	V_CORSAIR_STR, 289
P_HARPOON_STR, 284	vendor_str, 316
P_K65, 284	usb_add_device

usb_linux.c, 332	utf16to8
usb linux.c	profile.c, 220
_nk95cmd, 319	utf8to16
kbsyspath, 339	profile.c, 220
models, 339	, , , ,
N_MODELS, 318	V CORSAIR
os_closeusb, 320	usb.h, 289
os_inputmain, 321	V CORSAIR STR
os resetusb, 324	usb.h, 289
os_sendindicators, 325	vendor_str
os_setupusb, 326	usb.c, 276
os_usbrecv, 327	usb.h, 316
os_usbsend, 328	vtable_keyboard
strtrim, 330	device_vtable.c, 46
TEST_RESET, 318	structures.h, 258
udev, 340	vtable_keyboard_nonrgb
udev_enum, 331	device vtable.c, 46
udevthread, 340	structures.h, 258
usb_add_device, 332	vtable_mouse
usb_rm_device, 333	device_vtable.c, 46
usbadd, 334	structures.h, 258
usbclaim, 336	
usbkill, 336	
usbmain, 337	
usbthread, 340	
usbunclaim, 338	
usb_rm_device	
usb_linux.c, 333	
usb_tryreset usb.c, 275	
usb.h, 312	
usbadd	
usb_linux.c, 334 usbclaim	
usb_linux.c, 336	
usbdevice, 252	
usbid, 242 usbinput, 251	
usbkill	
usb.h, 314	
usb_linux.c, 336 usbmain	
usb.h, 314	
usb_linux.c, 337	
usbmode, 248	
usbmutex	
usb.c, 277	
usbprofile, 249	
usbrecv	
usb.h, 288	
usbsend	
usb.h, 289	
usbthread	
usb_linux.c, 340	
usbunclaim	
usb_linux.c, 338	
ushort	
includes.h, 89	
,	