

discoverpixy

Generated by Doxygen 1.8.9.1

Mon Jun 8 2015 11:01:19

Contents

1	discoverpixy Doxygen Documentation	1
2	Module Index	3
2.1	Modules	3
3	Data Structure Index	5
3.1	Data Structures	5
4	File Index	7
4.1	File List	7
5	Module Documentation	9
5.1	Application	9
5.1.1	Detailed Description	9
5.1.2	Function Documentation	9
5.1.2.1	app_init	10
5.1.2.2	app_process	10
5.2	Pixy Control Helper	12
5.2.1	Detailed Description	12
5.2.2	Function Documentation	12
5.2.2.1	pixy_PID_X	12
5.2.2.2	pixy_PID_Y	13
5.3	Pixy Frame Helper	14
5.3.1	Detailed Description	14
5.3.2	Function Documentation	14
5.3.2.1	pixy_cc_set_region	14
5.3.2.2	pixy_render_cropped_frame	15
5.3.2.3	pixy_render_full_frame	16
5.3.2.4	pixy_save_cropped_frame	17
5.3.2.5	pixy_save_full_frame	18
5.4	Filetest (Screen)	20
5.4.1	Detailed Description	20
5.4.2	Function Documentation	20

5.4.2.1	get_screen_filetest	20
5.5	Guitest (Screen)	21
5.5.1	Detailed Description	21
5.5.2	Function Documentation	21
5.5.2.1	get_screen_guitest	21
5.6	Screens	22
5.6.1	Detailed Description	22
5.7	Main (Screen)	23
5.7.1	Detailed Description	23
5.7.2	Function Documentation	23
5.7.2.1	get_screen_main	23
5.8	Photo Mode (Screen)	24
5.8.1	Detailed Description	24
5.8.2	Function Documentation	24
5.8.2.1	get_screen_photomode	24
5.9	Photo Mode Save (Screen)	25
5.9.1	Detailed Description	25
5.9.2	Function Documentation	25
5.9.2.1	get_screen_photomodesave	25
5.10	Pixytest (Screen)	26
5.10.1	Detailed Description	26
5.10.2	Function Documentation	26
5.10.2.1	get_screen_pixytest	26
5.11	Tracking (Screen)	27
5.11.1	Detailed Description	27
5.11.2	Enumeration Type Documentation	27
5.11.2.1	Tracking_Implementation	27
5.11.3	Function Documentation	27
5.11.3.1	get_screen_tracking	27
5.11.3.2	tracking_set_mode	28
5.12	Filesystem	29
5.12.1	Detailed Description	29
5.12.2	Enumeration Type Documentation	29
5.12.2.1	FILE_ATTRIBUTES	29
5.12.2.2	FILE_STATUS	30
5.12.3	Function Documentation	30
5.12.3.1	filesystem_dir_close	30
5.12.3.2	filesystem_dir_open	31
5.12.3.3	filesystem_file_close	32
5.12.3.4	filesystem_file_open	33

5.12.3.5	filesystem_file_read	33
5.12.3.6	filesystem_file_seek	34
5.12.3.7	filesystem_file_write	35
5.12.3.8	filesystem_init	36
5.13	Gui	38
5.13.1	Detailed Description	38
5.14	Button	39
5.14.1	Detailed Description	39
5.14.2	Macro Definition Documentation	39
5.14.2.1	AUTO	39
5.14.3	Typedef Documentation	39
5.14.3.1	BUTTON_CALLBACK	39
5.14.4	Function Documentation	40
5.14.4.1	gui_button_add	40
5.14.4.2	gui_button_redraw	41
5.14.4.3	gui_button_remove	42
5.15	Checkbox	44
5.15.1	Detailed Description	44
5.15.2	Macro Definition Documentation	44
5.15.2.1	CHECKBOX_WIN_FG_COLOR	44
5.15.3	Typedef Documentation	44
5.15.3.1	CHECKBOX_CALLBACK	44
5.15.4	Function Documentation	45
5.15.4.1	gui_checkbox_add	45
5.15.4.2	gui_checkbox_redraw	46
5.15.4.3	gui_checkbox_remove	47
5.15.4.4	gui_checkbox_update	47
5.16	NumericUpDown	49
5.16.1	Detailed Description	49
5.16.2	Typedef Documentation	49
5.16.2.1	NUMUPDOWN_CALLBACK	49
5.16.3	Function Documentation	50
5.16.3.1	gui_numupdown_add	50
5.16.3.2	gui_numupdown_redraw	51
5.16.3.3	gui_numupdown_remove	52
5.16.3.4	gui_numupdown_update	53
5.17	Screen	55
5.17.1	Detailed Description	55
5.17.2	Typedef Documentation	55
5.17.2.1	SCREEN_CALLBACK	55

5.17.2.2	SCREEN_STRUCT	55
5.17.3	Function Documentation	56
5.17.3.1	gui_screen_back	56
5.17.3.2	gui_screen_get_current	56
5.17.3.3	gui_screen_navigate	57
5.17.3.4	gui_screen_update	58
5.18	Filesystem (LowLevel)	59
5.18.1	Detailed Description	59
5.18.2	Function Documentation	59
5.18.2.1	ll_filesystem_dir_close	59
5.18.2.2	ll_filesystem_dir_open	60
5.18.2.3	ll_filesystem_file_close	60
5.18.2.4	ll_filesystem_file_open	60
5.18.2.5	ll_filesystem_file_read	61
5.18.2.6	ll_filesystem_file_seek	61
5.18.2.7	ll_filesystem_file_write	61
5.18.2.8	ll_filesystem_init	61
5.19	LowLevel	62
5.19.1	Detailed Description	62
5.20	System (LowLevel)	63
5.20.1	Detailed Description	63
5.20.2	Function Documentation	63
5.20.2.1	ll_system_delay	63
5.20.2.2	ll_system_init	64
5.20.2.3	ll_system_process	64
5.20.2.4	ll_system_toggle_led	64
5.21	TFT (LowLevel)	65
5.21.1	Detailed Description	65
5.21.2	Function Documentation	65
5.21.2.1	ll_tft_clear	65
5.21.2.2	ll_tft_draw_bitmap_unscaled	66
5.21.2.3	ll_tft_draw_char	66
5.21.2.4	ll_tft_draw_circle	66
5.21.2.5	ll_tft_draw_line	67
5.21.2.6	ll_tft_draw_pixel	67
5.21.2.7	ll_tft_draw_rectangle	67
5.21.2.8	ll_tft_fill_rectangle	68
5.21.2.9	ll_tft_font_height	68
5.21.2.10	ll_tft_font_width	68
5.21.2.11	ll_tft_init	69

5.21.2.12 ll_tft_num_fonts	69
5.22 Touch (LowLevel)	70
5.22.1 Detailed Description	70
5.22.2 Function Documentation	70
5.22.2.1 ll_touch_init	70
5.23 Pixy	71
5.23.1 Detailed Description	72
5.23.2 Macro Definition Documentation	72
5.23.2.1 PIXY_BLOCKTYPE_COLOR_CODE	72
5.23.2.2 PIXY_BLOCKTYPE_NORMAL	72
5.23.2.3 PIXY_MAX_SIGNATURE	72
5.23.2.4 PIXY_MAX_X	72
5.23.2.5 PIXY_MAX_Y	72
5.23.2.6 PIXY_MIN_X	72
5.23.2.7 PIXY_MIN_Y	73
5.23.2.8 PIXY_RCS_CENTER_POS	73
5.23.2.9 PIXY_RCS_MAX_POS	73
5.23.2.10 PIXY_RCS_MIN_POS	73
5.23.3 Function Documentation	73
5.23.3.1 pixy_blocks_are_new	73
5.23.3.2 pixy_cam_get_auto_exposure_compensation	73
5.23.3.3 pixy_cam_get_auto_white_balance	74
5.23.3.4 pixy_cam_get_brightness	74
5.23.3.5 pixy_cam_get_exposure_compensation	74
5.23.3.6 pixy_cam_get_white_balance_value	74
5.23.3.7 pixy_cam_set_auto_exposure_compensation	74
5.23.3.8 pixy_cam_set_auto_white_balance	75
5.23.3.9 pixy_cam_set_brightness	76
5.23.3.10 pixy_cam_set_exposure_compensation	76
5.23.3.11 pixy_cam_set_white_balance_value	76
5.23.3.12 pixy_close	76
5.23.3.13 pixy_command	77
5.23.3.14 pixy_error	77
5.23.3.15 pixy_get_blocks	78
5.23.3.16 pixy_get_firmware_version	78
5.23.3.17 pixy_init	78
5.23.3.18 pixy_led_get_max_current	79
5.23.3.19 pixy_led_set_max_current	79
5.23.3.20 pixy_led_set_RGB	80
5.23.3.21 pixy_rcs_get_position	81

5.23.3.22	pixy_rcs_set_frequency	81
5.23.3.23	pixy_rcs_set_position	81
5.23.3.24	pixy_service	82
5.24	System	83
5.24.1	Detailed Description	83
5.24.2	Function Documentation	83
5.24.2.1	system_delay	83
5.24.2.2	system_init	83
5.24.2.3	system_process	84
5.24.2.4	system_toggle_led	85
5.25	TFT	86
5.25.1	Detailed Description	86
5.25.2	Macro Definition Documentation	86
5.25.2.1	BLACK	86
5.25.2.2	BLUE	86
5.25.2.3	GREEN	86
5.25.2.4	HEX	87
5.25.2.5	RED	87
5.25.2.6	RGB	87
5.25.2.7	TRANSPARENT	87
5.25.2.8	WHITE	87
5.25.3	Function Documentation	87
5.25.3.1	tft_clear	87
5.25.3.2	tft_draw_bitmap_file_unscaled	88
5.25.3.3	tft_draw_bitmap_unscaled	90
5.25.3.4	tft_draw_circle	91
5.25.3.5	tft_draw_line	92
5.25.3.6	tft_draw_pixel	93
5.25.3.7	tft_draw_rectangle	94
5.25.3.8	tft_fill_rectangle	95
5.25.3.9	tft_font_height	96
5.25.3.10	tft_font_width	97
5.25.3.11	tft_init	98
5.25.3.12	tft_num_fonts	99
5.25.3.13	tft_print_formatted	99
5.25.3.14	tft_print_line	100
5.26	Calibrate (Screen)	102
5.26.1	Detailed Description	102
5.26.2	Function Documentation	102
5.26.2.1	get_screen_calibrate	102

5.27 Touch	103
5.27.1 Detailed Description	103
5.27.2 Typedef Documentation	104
5.27.2.1 TOUCH_CALLBACK	104
5.27.3 Enumeration Type Documentation	104
5.27.3.1 TOUCH_ACTION	104
5.27.3.2 TOUCH_STATE	104
5.27.4 Function Documentation	105
5.27.4.1 touch_add_raw_event	105
5.27.4.2 touch_get_last_point	106
5.27.4.3 touch_have_empty	107
5.27.4.4 touch_init	107
5.27.4.5 touch_register_area	108
5.27.4.6 touch_set_calibration_values	109
5.27.4.7 touch_set_value_convert_mode	109
5.27.4.8 touch_unregister_area	110
6 Data Structure Documentation	111
6.1 Block Struct Reference	111
6.1.1 Detailed Description	111
6.1.2 Field Documentation	111
6.1.2.1 angle	111
6.1.2.2 height	111
6.1.2.3 signature	111
6.1.2.4 type	111
6.1.2.5 width	112
6.1.2.6 x	112
6.1.2.7 y	112
6.2 BUTTON_STRUCT Struct Reference	112
6.2.1 Detailed Description	113
6.2.2 Field Documentation	113
6.2.2.1 base	113
6.2.2.2 bgcolor	113
6.2.2.3 callback	113
6.2.2.4 font	113
6.2.2.5 text	113
6.2.2.6 txtcolor	113
6.3 CHECKBOX_STRUCT Struct Reference	113
6.3.1 Detailed Description	114
6.3.2 Field Documentation	114

6.3.2.1	base	114
6.3.2.2	callback	114
6.3.2.3	checked	114
6.3.2.4	fgcolor	115
6.4	DIRECTORY_STRUCT Struct Reference	115
6.4.1	Detailed Description	115
6.4.2	Field Documentation	116
6.4.2.1	files	116
6.4.2.2	num_files	116
6.4.2.3	path	116
6.5	FILE_DATE_STRUCT Struct Reference	116
6.5.1	Detailed Description	116
6.5.2	Field Documentation	116
6.5.2.1	day	116
6.5.2.2	month	117
6.5.2.3	year	117
6.6	FILE_HANDLE Struct Reference	117
6.6.1	Detailed Description	117
6.6.2	Field Documentation	117
6.6.2.1	fname	117
6.6.2.2	fpos	117
6.6.2.3	fsize	118
6.7	FILE_LIST_ENTRY_S Struct Reference	118
6.7.1	Detailed Description	118
6.7.2	Field Documentation	118
6.7.2.1	filename	118
6.7.2.2	next	118
6.8	FILE_STRUCT Struct Reference	118
6.8.1	Detailed Description	119
6.8.2	Field Documentation	119
6.8.2.1	fattrib	119
6.8.2.2	fdate	119
6.8.2.3	fname	120
6.8.2.4	fsize	120
6.8.2.5	ftime	120
6.9	FILE_TIME_STRUCT Struct Reference	120
6.9.1	Detailed Description	120
6.9.2	Field Documentation	120
6.9.2.1	hour	120
6.9.2.2	min	120

6.9.2.3	sec	121
6.10	NUMUPDOWN_STRUCT Struct Reference	121
6.10.1	Detailed Description	122
6.10.2	Field Documentation	122
6.10.2.1	buttonDown	122
6.10.2.2	buttonUp	122
6.10.2.3	callback	122
6.10.2.4	fgcolor	122
6.10.2.5	max	122
6.10.2.6	min	122
6.10.2.7	value	122
6.10.2.8	x	123
6.10.2.9	y	123
6.11	POINT_STRUCT Struct Reference	123
6.11.1	Detailed Description	123
6.11.2	Field Documentation	123
6.11.2.1	x	123
6.11.2.2	y	123
6.12	SCREEN_S Struct Reference	124
6.12.1	Detailed Description	124
6.12.2	Field Documentation	124
6.12.2.1	next	124
6.12.2.2	on_enter	124
6.12.2.3	on_leave	124
6.12.2.4	on_update	125
6.13	TOUCH_AREA_STRUCT Struct Reference	125
6.13.1	Detailed Description	125
6.13.2	Field Documentation	125
6.13.2.1	callback	125
6.13.2.2	flags	125
6.13.2.3	hookedActions	126
6.13.2.4	x1	126
6.13.2.5	x2	126
6.13.2.6	y1	126
6.13.2.7	y2	126
6.14	TRACKING_CONFIG_STRUCT Struct Reference	126
6.14.1	Detailed Description	127
6.14.2	Field Documentation	127
6.14.2.1	start	127
6.14.2.2	stop	127

6.14.2.3	update	127
7	File Documentation	129
7.1	common/app/app.c File Reference	129
7.2	app.c	129
7.3	common/app/app.h File Reference	131
7.4	app.h	131
7.5	common/app/pixy_control.c File Reference	131
7.5.1	Macro Definition Documentation	132
7.5.1.1	REG_PID_KD	132
7.5.1.2	REG_PID_KI	132
7.5.1.3	REG_PID_KP	132
7.5.1.4	REG_PID_TA	132
7.6	pixy_control.c	133
7.7	common/app/pixy_control.h File Reference	134
7.8	pixy_control.h	134
7.9	common/app/pixy_frame.c File Reference	135
7.9.1	Function Documentation	136
7.9.1.1	interpolateBayer	136
7.9.1.2	renderBA81	136
7.9.1.3	saveBA81	138
7.10	pixy_frame.c	139
7.11	common/app/pixy_frame.h File Reference	142
7.12	pixy_frame.h	143
7.13	common/app/screen_filetest.c File Reference	143
7.13.1	Function Documentation	144
7.13.1.1	b_back_cb	144
7.13.1.2	enter	145
7.13.1.3	image_test	147
7.13.1.4	leave	148
7.13.1.5	update	149
7.13.2	Variable Documentation	149
7.13.2.1	b_back	149
7.13.2.2	screen	149
7.14	screen_filetest.c	149
7.15	common/app/screen_filetest.h File Reference	151
7.16	screen_filetest.h	152
7.17	common/app/screen_guitest.c File Reference	152
7.17.1	Function Documentation	153
7.17.1.1	b_back_cb	153

7.17.1.2	checkboxCB	154
7.17.1.3	enter	154
7.17.1.4	leave	156
7.17.1.5	n_updown_cb	157
7.17.1.6	touchCB	157
7.17.1.7	update	158
7.17.2	Variable Documentation	158
7.17.2.1	a_area	158
7.17.2.2	b_back	158
7.17.2.3	c_cbox	158
7.17.2.4	n_updown	158
7.17.2.5	screen	158
7.18	screen_guitest.c	158
7.19	common/app/screen_guitest.h File Reference	161
7.20	screen_guitest.h	161
7.21	common/app/screen_main.c File Reference	162
7.21.1	Macro Definition Documentation	163
7.21.1.1	BUTTON_SPACING	163
7.21.1.2	X_TAB	163
7.21.1.3	Y_FIRST	163
7.21.1.4	Y_SECOND	163
7.21.1.5	Y_THIRD	163
7.21.2	Function Documentation	163
7.21.2.1	b_filetest_cb	163
7.21.2.2	b_guitest_cb	164
7.21.2.3	b_our_tracking_cb	164
7.21.2.4	b_photo_mode_cb	165
7.21.2.5	b_pixytest_cb	166
7.21.2.6	b_ref_tracking_cb	166
7.21.2.7	enter	167
7.21.2.8	leave	169
7.21.2.9	update	170
7.21.3	Variable Documentation	170
7.21.3.1	b_filetest	170
7.21.3.2	b_guitest	170
7.21.3.3	b_our_tracking	170
7.21.3.4	b_photo_mode	170
7.21.3.5	b_pixytest	170
7.21.3.6	b_ref_tracking	170
7.21.3.7	screen	170

7.22	screen_main.c	171
7.23	common/app/screen_main.h File Reference	173
7.24	screen_main.h	174
7.25	common/app/screen_photomode.c File Reference	174
7.25.1	Function Documentation	175
7.25.1.1	b_back_cb	175
7.25.1.2	b_save_cb	176
7.25.1.3	enter	177
7.25.1.4	leave	178
7.25.1.5	touchCB	178
7.25.1.6	update	180
7.25.2	Variable Documentation	180
7.25.2.1	a_area	180
7.25.2.2	b_back	181
7.25.2.3	b_save	181
7.25.2.4	old_pos	181
7.25.2.5	pixy_connected	181
7.25.2.6	pixy_pos	181
7.25.2.7	screen	181
7.25.2.8	subMenu	181
7.26	screen_photomode.c	181
7.27	common/app/screen_photomode.h File Reference	184
7.28	screen_photomode.h	185
7.29	common/app/screen_photomode_save.c File Reference	185
7.29.1	Macro Definition Documentation	186
7.29.1.1	X_OFS	186
7.29.2	Typedef Documentation	186
7.29.2.1	FILE_LIST_ENTRY	186
7.29.3	Enumeration Type Documentation	186
7.29.3.1	anonymous enum	186
7.29.4	Function Documentation	187
7.29.4.1	b_back_cb	187
7.29.4.2	enter	187
7.29.4.3	leave	188
7.29.4.4	touchCB	189
7.29.4.5	update	190
7.29.5	Variable Documentation	193
7.29.5.1	a_area	193
7.29.5.2	b_back	193
7.29.5.3	bmpheader_data	193

7.29.5.4	files_ok	193
7.29.5.5	fontheight	193
7.29.5.6	liststart	193
7.29.5.7	nomatch_text	193
7.29.5.8	num_files_ok	194
7.29.5.9	picked_file	194
7.29.5.10	screen	194
7.29.5.11	state	194
7.30	screen_photomode_save.c	194
7.31	common/app/screen_photomode_save.h File Reference	199
7.32	screen_photomode_save.h	199
7.33	common/app/screen_pixytest.c File Reference	200
7.33.1	Macro Definition Documentation	201
7.33.1.1	LED_COLOR_BUTTON_SPACING	201
7.33.1.2	LED_COLOR_BUTTON_Y	201
7.33.1.3	LED_POWER_BUTTON_Y	201
7.33.1.4	SERVO_BUTTON_SPACING	201
7.33.1.5	SERVO_BUTTON_Y	201
7.33.2	Enumeration Type Documentation	201
7.33.2.1	anonymous enum	201
7.33.3	Function Documentation	202
7.33.3.1	b_back_cb	202
7.33.3.2	b_led_blue_cb	202
7.33.3.3	b_led_green_cb	203
7.33.3.4	b_led_off_cb	203
7.33.3.5	b_led_red_cb	204
7.33.3.6	b_led_white_cb	204
7.33.3.7	b_servos_bottomleft_cb	204
7.33.3.8	b_servos_bottomright_cb	205
7.33.3.9	b_servos_center_cb	205
7.33.3.10	b_servos_topleft_cb	206
7.33.3.11	b_servos_topright_cb	206
7.33.3.12	enter	206
7.33.3.13	leave	209
7.33.3.14	n_led_powerlimit_cb	209
7.33.3.15	update	210
7.33.4	Variable Documentation	211
7.33.4.1	b_back	211
7.33.4.2	b_led_blue	211
7.33.4.3	b_led_green	211

7.33.4.4	b_led_off	211
7.33.4.5	b_led_red	211
7.33.4.6	b_led_white	211
7.33.4.7	b_servos_bottomleft	212
7.33.4.8	b_servos_bottomright	212
7.33.4.9	b_servos_center	212
7.33.4.10	b_servos_topleft	212
7.33.4.11	b_servos_topright	212
7.33.4.12	led_color	212
7.33.4.13	led_maxcurrent	212
7.33.4.14	n_led_powerlimit	212
7.33.4.15	screen	212
7.33.4.16	servo_x	212
7.33.4.17	servo_y	212
7.33.4.18	state	213
7.34	screen_pixytest.c	213
7.35	common/app/screen_pixytest.h File Reference	218
7.36	screen_pixytest.h	218
7.37	common/app/screen_tracking.c File Reference	219
7.37.1	Macro Definition Documentation	220
7.37.1.1	BLOCK_BUFFER_SIZE	220
7.37.1.2	FRAME_END_X	220
7.37.1.3	FRAME_END_Y	220
7.37.1.4	FRAME_HEIGHT	221
7.37.1.5	FRAME_START_X	221
7.37.1.6	FRAME_START_Y	221
7.37.1.7	FRAME_WIDTH	221
7.37.2	Typedef Documentation	221
7.37.2.1	TRACKING_BLOCK_CALLBACK	221
7.37.2.2	TRACKING_VOID_CALLBACK	221
7.37.3	Enumeration Type Documentation	221
7.37.3.1	anonymous enum	221
7.37.4	Function Documentation	222
7.37.4.1	b_back_cb	222
7.37.4.2	b_select_cb	222
7.37.4.3	c_frame_toggle_cb	223
7.37.4.4	enter	223
7.37.4.5	leave	224
7.37.4.6	touchCB	225
7.37.4.7	tracking_our_start	226

7.37.4.8	tracking_our_stop	226
7.37.4.9	tracking_our_update	227
7.37.4.10	tracking_reference_start	228
7.37.4.11	tracking_reference_stop	228
7.37.4.12	tracking_reference_update	228
7.37.4.13	update	229
7.37.5	Variable Documentation	230
7.37.5.1	a_area	230
7.37.5.2	b_back	230
7.37.5.3	b_select	231
7.37.5.4	c_frame_toggle	231
7.37.5.5	frame_visible	231
7.37.5.6	point1	231
7.37.5.7	point1_valid	231
7.37.5.8	point2	231
7.37.5.9	screen	231
7.37.5.10	servo_x	231
7.37.5.11	servo_y	231
7.37.5.12	state	231
7.37.5.13	tracking_current	231
7.37.5.14	tracking_our	232
7.37.5.15	tracking_reference	232
7.38	screen_tracking.c	232
7.39	common/app/screen_tracking.h File Reference	238
7.40	screen_tracking.h	238
7.41	common/filesystem/filesystem.c File Reference	239
7.42	filesystem.c	240
7.43	common/filesystem/filesystem.h File Reference	240
7.44	filesystem.h	242
7.45	common/gui/button.c File Reference	243
7.45.1	Macro Definition Documentation	244
7.45.1.1	BRIGHTNESS_VAL	244
7.45.2	Function Documentation	244
7.45.2.1	buttons_cb	244
7.45.2.2	calculate_shadows	245
7.46	button.c	246
7.47	common/gui/button.h File Reference	249
7.48	button.h	250
7.49	common/gui/checkbox.c File Reference	250
7.49.1	Macro Definition Documentation	251

7.49.1.1	ACTIVE_COLOR	251
7.49.1.2	BACKGROUND_COLOR	251
7.49.1.3	BORDER_COLOR	251
7.49.2	Function Documentation	252
7.49.2.1	checkboxes_cb	252
7.50	checkbox.c	253
7.51	common/gui/checkbox.h File Reference	255
7.52	checkbox.h	256
7.53	common/gui/numupdown.c File Reference	256
7.53.1	Macro Definition Documentation	257
7.53.1.1	BASE_COLOR	257
7.53.2	Function Documentation	257
7.53.2.1	button_down_cb	257
7.53.2.2	button_up_cb	258
7.53.2.3	calc_text_width	259
7.54	numupdown.c	259
7.55	common/gui/numupdown.h File Reference	262
7.56	numupdown.h	263
7.57	common/gui/screen.c File Reference	264
7.57.1	Variable Documentation	264
7.57.1.1	screen_current	264
7.57.1.2	screen_goto	264
7.57.1.3	screen_list	264
7.58	screen.c	265
7.59	common/gui/screen.h File Reference	266
7.60	screen.h	267
7.61	common/lowlevel/ll_filesystem.h File Reference	268
7.62	ll_filesystem.h	269
7.63	common/lowlevel/ll_system.h File Reference	269
7.64	ll_system.h	270
7.65	common/lowlevel/ll_tft.h File Reference	271
7.66	ll_tft.h	272
7.67	common/lowlevel/ll_touch.h File Reference	272
7.68	ll_touch.h	273
7.69	common/pixy/pixy.h File Reference	274
7.70	pixy.h	276
7.71	common/pixy/pixydefs.h File Reference	277
7.71.1	Macro Definition Documentation	278
7.71.1.1	CRP_ARRAY	278
7.71.1.2	CRP_FLT	279

7.71.1.3	CRP_FLT32	279
7.71.1.4	CRP_FLT64	279
7.71.1.5	CRP_FLTS32	279
7.71.1.6	CRP_FLTS64	279
7.71.1.7	CRP_INT16	279
7.71.1.8	CRP_INT32	279
7.71.1.9	CRP_INT8	279
7.71.1.10	CRP_INTS16	279
7.71.1.11	CRP_INTS32	279
7.71.1.12	CRP_INTS8	279
7.71.1.13	CRP_NO_COPY	279
7.71.1.14	CRP_NULLTERM_ARRAY	280
7.71.1.15	CRP_STRING	280
7.71.1.16	CRP_TYPE_HINT	280
7.71.1.17	CRP_UINT16	280
7.71.1.18	CRP_UINT32	280
7.71.1.19	CRP_UINT8	280
7.71.1.20	CRP_UINTS16	280
7.71.1.21	CRP_UINTS16_NO_COPY	280
7.71.1.22	CRP_UINTS32	280
7.71.1.23	CRP_UINTS32_NO_COPY	280
7.71.1.24	CRP_UINTS8	280
7.71.1.25	CRP_UINTS8_NO_COPY	280
7.71.1.26	END	281
7.71.1.27	END_IN_ARGS	281
7.71.1.28	END_OUT_ARGS	281
7.71.1.29	FLT32	281
7.71.1.30	FLT64	281
7.71.1.31	FLTS32	281
7.71.1.32	FLTS64	281
7.71.1.33	INT16	281
7.71.1.34	INT32	281
7.71.1.35	INT8	281
7.71.1.36	INTS16	281
7.71.1.37	INTS32	281
7.71.1.38	INTS8	282
7.71.1.39	PIXY_ERROR_CHIRP	282
7.71.1.40	PIXY_ERROR_INVALID_COMMAND	282
7.71.1.41	PIXY_ERROR_INVALID_PARAMETER	282
7.71.1.42	STRING	282

7.71.1.43	UINT16	282
7.71.1.44	UINT32	282
7.71.1.45	UINT8	282
7.71.1.46	UINTS16	282
7.71.1.47	UINTS16_NO_COPY	282
7.71.1.48	UINTS32	282
7.71.1.49	UINTS32_NO_COPY	282
7.71.1.50	UINTS8	283
7.71.1.51	UINTS8_NO_COPY	283
7.72	pixydefs.h	283
7.73	common/system/system.c File Reference	284
7.74	system.c	284
7.75	common/system/system.h File Reference	285
7.76	system.h	286
7.77	common/tft/tft.c File Reference	286
7.78	tft.c	287
7.79	common/tft/tft.h File Reference	290
7.80	tft.h	291
7.81	common/touch/screen_calibrate.c File Reference	292
7.81.1	Function Documentation	293
7.81.1.1	enter	293
7.81.1.2	leave	293
7.81.1.3	update	293
7.81.2	Variable Documentation	295
7.81.2.1	calibration	295
7.81.2.2	screen	295
7.82	screen_calibrate.c	295
7.83	common/touch/screen_calibrate.h File Reference	297
7.83.1	Macro Definition Documentation	298
7.83.1.1	CBEGIN	298
7.83.1.2	CCENTER	298
7.83.1.3	CEND	298
7.83.1.4	CLENGTH	298
7.83.1.5	DHEIGHT	298
7.83.1.6	DWIDTH	299
7.84	screen_calibrate.h	299
7.85	common/touch/touch.c File Reference	299
7.85.1	Macro Definition Documentation	301
7.85.1.1	NUM_AREAS	301
7.85.2	Variable Documentation	301

7.85.2.1	areas	301
7.85.2.2	cal_dx	301
7.85.2.3	cal_dy	301
7.85.2.4	cal_xs	301
7.85.2.5	cal_ys	301
7.85.2.6	calibration	301
7.85.2.7	oldState	301
7.85.2.8	pos	301
7.85.2.9	use_calibration	301
7.86	touch.c	301
7.87	common/touch/touch.h File Reference	304
7.88	touch.h	306

Chapter 1

discoverpixy Doxygen Documentation

Welcome to the code-documentation for all common (and platformindependent) code.
A good point to start is probably the [Modules](#) page.

Chapter 2

Module Index

2.1 Modules

Here is a list of all modules:

Application	9
Pixy Control Helper	12
Pixy Frame Helper	14
Screens	22
Filetest (Screen)	20
Guitest (Screen)	21
Main (Screen)	23
Photo Mode (Screen)	24
Photo Mode Save (Screen)	25
Pixytest (Screen)	26
Tracking (Screen)	27
Filesystem	29
Gui	38
Button	39
Checkbox	44
NumericUpDown	49
Screen	55
LowLevel	62
Filesystem (LowLevel)	59
System (LowLevel)	63
TFT (LowLevel)	65
Touch (LowLevel)	70
Pixy	71
System	83
TFT	86
Touch	103
Calibrate (Screen)	102

Chapter 3

Data Structure Index

3.1 Data Structures

Here are the data structures with brief descriptions:

Block	111
BUTTON_STRUCT	112
CHECKBOX_STRUCT	113
DIRECTORY_STRUCT	115
FILE_DATE_STRUCT	116
FILE_HANDLE	117
FILE_LIST_ENTRY_S	118
FILE_STRUCT	118
FILE_TIME_STRUCT	120
NUMUPDOWN_STRUCT	121
POINT_STRUCT	123
SCREEN_S	124
TOUCH_AREA_STRUCT	125
TRACKING_CONFIG_STRUCT	126

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

common/app/app.c	129
common/app/app.h	131
common/app/pixy_control.c	131
common/app/pixy_control.h	134
common/app/pixy_frame.c	135
common/app/pixy_frame.h	142
common/app/screen_filetest.c	143
common/app/screen_filetest.h	151
common/app/screen_guitest.c	152
common/app/screen_guitest.h	161
common/app/screen_main.c	162
common/app/screen_main.h	173
common/app/screen_photomode.c	174
common/app/screen_photomode.h	184
common/app/screen_photomode_save.c	185
common/app/screen_photomode_save.h	199
common/app/screen_pixytest.c	200
common/app/screen_pixytest.h	218
common/app/screen_tracking.c	219
common/app/screen_tracking.h	238
common/filesystem/filesystem.c	239
common/filesystem/filesystem.h	240
common/gui/button.c	243
common/gui/button.h	249
common/gui/checkbox.c	250
common/gui/checkbox.h	255
common/gui/numupdown.c	256
common/gui/numupdown.h	262
common/gui/screen.c	264
common/gui/screen.h	266
common/lowlevel/ll_filesystem.h	268
common/lowlevel/ll_system.h	269
common/lowlevel/ll_tft.h	271
common/lowlevel/ll_touch.h	272
common/pixy/pixy.h	274
common/pixy/pixydefs.h	277
common/system/system.c	284
common/system/system.h	285

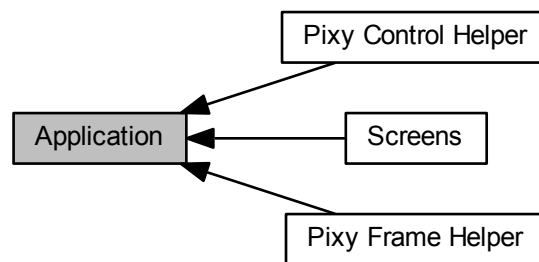
common/tft/tft.c	286
common/tft/tft.h	290
common/touch/screen_calibrate.c	292
common/touch/screen_calibrate.h	297
common/touch/touch.c	299
common/touch/touch.h	304

Chapter 5

Module Documentation

5.1 Application

Collaboration diagram for Application:



Modules

- [Pixy Control Helper](#)
- [Pixy Frame Helper](#)
- [Screens](#)

Functions

- void [app_init](#) ()
- void [app_process](#) ()

5.1.1 Detailed Description

The App Module contains the effective, platform independent application.

5.1.2 Function Documentation

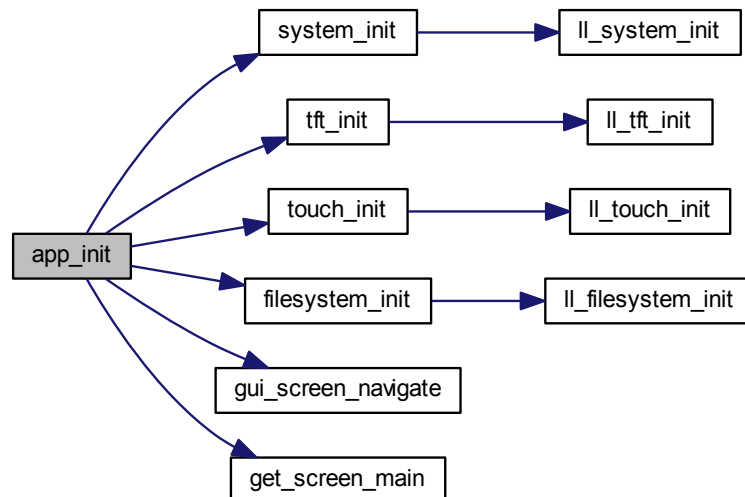
5.1.2.1 void app_init ()

Starts/Initializes the app This function should be called at the top of the main function of your platform

Definition at line 37 of file [app.c](#).

```
00038 {
00039     system_init();
00040     tft_init();
00041     touch_init();
00042     filesystem_init();
00043
00044     gui_screen_navigate(get_screen_main());
00045 }
```

Here is the call graph for this function:



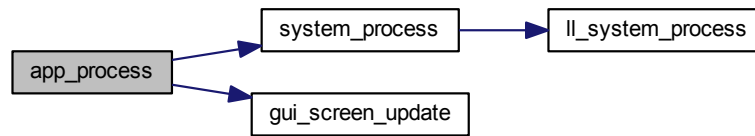
5.1.2.2 void app_process ()

Executes one cycle of the app Call this function repeatedly from a loop inside the main function

Definition at line 48 of file [app.c](#).

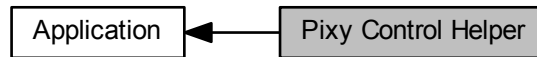
```
00049 {
00050
00051     system_process(); //Let the system handle it's pending events
00052     gui_screen_update(); //update the currently active screen
00053 }
```


Here is the call graph for this function:



5.2 Pixy Control Helper

Collaboration diagram for Pixy Control Helper:



Functions

- `int16_t pixy_PID_Y (int16_t x, int16_t w)`
- `int16_t pixy_PID_X (int16_t x, int16_t w)`

5.2.1 Detailed Description

A collection of helper functions that contain PID Control code used for object tracking.

5.2.2 Function Documentation

5.2.2.1 `int16_t pixy_PID_X ( int16_t x, int16_t w )`

Execute one step of PID regulation for the X-servo.

Parameters

<code>x</code>	desired value (e.g. current x-position of the biggest block)
<code>w</code>	actual value (e.g. desired x-position)

Returns

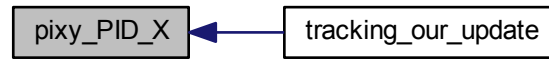
The offset which needs to be added to the X-Servo position

Definition at line 67 of file `pixy_control.c`.

```

00068 {
00069     float e = 0;
00070     static float esum = 0;
00071     static float eold = 0;
00072     float y = 0;
00073
00074     e = (float)(x - w); // calculate the controller offset
00075
00076     //----PID-control-----
00077     esum = esum + e; // add e to the current sum
00078
00079     y += REG_PID_KP * e; // add the proportional part to the output
00080     y += REG_PID_KI * REG_PID_TA * esum; // add the integral part to the output
00081     y += REG_PID_KD * (e - eold) / REG_PID_TA; // add the differential part to the
output
00082     //-----
00083
00084     eold = e; // save the previous value
00085
00086     return (int16_t)y;
00087 }
  
```

Here is the caller graph for this function:



5.2.2.2 int16_t pixy_PID_Y (int16_t x, int16_t w)

Execute one step of PID regulation for the Y-servo.

Parameters

x	desired value (e.g. current y-position of the biggest block)
w	actual value (e.g desired y-position)

Returns

The offset which needs to be added to the Y-Servo position

Definition at line 44 of file [pixy_control.c](#).

```

00045 {
00046     float e = 0;
00047     static float esum = 0;
00048     static float eold = 0;
00049     float y = 0;
00050
00051     e = (float)(x - w); // calculate the controller offset
00052
00053     //----PID-control-----
00054     esum = esum + e; // add e to the current sum
00055
00056     y += REG_PID_KP * e; // add the proportional part to the output
00057     y += REG_PID_KI * REG_PID_TA * esum; // add the integral part to the output
00058     y += REG_PID_KD * (e - eold) / REG_PID_TA; // add the differential part to the
output
00059     //-----
00060
00061     eold = e; // save the previous value
00062
00063     return (int16_t)y;
00064 }
  
```

Here is the caller graph for this function:



5.3 Pixy Frame Helper

Collaboration diagram for Pixy Frame Helper:



Functions

- int [pixy_render_full_frame](#) (uint16_t x, uint16_t y)
- int [pixy_render_cropped_frame](#) (uint16_t x, uint16_t y, uint16_t xoffset, uint16_t yoffset, uint16_t width, uint16_t height)
- int [pixy_save_full_frame](#) (FILE_HANDLE *handle)
- int [pixy_save_cropped_frame](#) (FILE_HANDLE *handle, uint16_t xoffset, uint16_t yoffset, uint16_t width, uint16_t height)
- int [pixy_cc_set_region](#) (uint8_t signum, uint16_t xoffset, uint16_t yoffset, uint16_t width, uint16_t height)

5.3.1 Detailed Description

A collection of helper functions that allow receiving and rendering a frame from pixy onto the display. Furthermore you can select a color in a frame, to use for tracking.

5.3.2 Function Documentation

5.3.2.1 int [pixy_cc_set_region](#) (uint8_t *signum*, uint16_t *xoffset*, uint16_t *yoffset*, uint16_t *width*, uint16_t *height*)

Sets the color signature to the color in the selected region of the frame

Parameters

<i>signum</i>	the color signature number (1..7)
<i>xoffset</i>	The x-Coordinate of the topleft point of the region
<i>yoffset</i>	The y-Coordinate of the topleft point of the region
<i>width</i>	The width of the region
<i>height</i>	The height of the region

Returns

0 on success, otherwise the errorcode from pixy

Definition at line 233 of file [pixy_frame.c](#).

```

00234 {
00235     int32_t response;
00236
00237     int return_value = pixy\_command("cc_setSigRegion", // String id for remote procedure
00238                                     INT32(0),          // type = normal color code
00239                                     INT8(signum),
00240                                     INT16(xoffset),      // xoffset
00241                                     INT16(yoffset),     // yoffset
00242                                     INT16(width),       // width
  
```

```

00243         INT16(height),           // height
00244         END_OUT_ARGS,           // separator
00245         &response,              // pointer to mem address for return value
00246         END_IN_ARGS);
00247     return return_value;
00248 }

```

Here is the call graph for this function:



Here is the caller graph for this function:



5.3.2.2 `int pixy_render_cropped_frame ( uint16_t x, uint16_t y, uint16_t xoffset, uint16_t yoffset, uint16_t width, uint16_t height )`

Receives a cropped frame from pixy and display's it on the display with the topleft corner at (x,y)

Parameters

<i>x</i>	The x-Coordinate of the top left corner to draw the image
<i>y</i>	The y-Coordinate of the top left corner to draw the image
<i>xoffset</i>	The x-Coordinate on the pixy image from where on you want the frame data
<i>yoffset</i>	The y-Coordinate on the pixy image from where on you want the frame data
<i>width</i>	The width of the image recorded from pixy
<i>height</i>	The height of the image recorded from pixy

Returns

0 on success, otherwise the errorcode from pixy

Definition at line 31 of file [pixy_frame.c](#).

```

00032 {
00033     uint8_t* videodata;
00034     int32_t response;
00035     int32_t fourccc;
00036     int8_t renderflags;
00037     uint16_t xwidth;
00038     uint16_t ywidth;
00039     uint32_t size;
00040

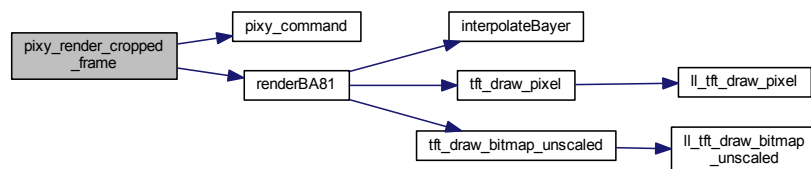
```

```

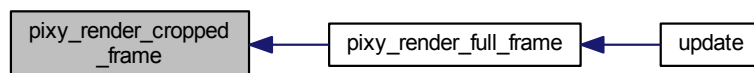
00041
00042     int return_value = pixy_command("cam_getFrame", // String id for remote procedure
00043                                     INT8(0x21), // mode
00044                                     INT16(xoffset), // xoffset
00045                                     INT16(yoffset), // yoffset
00046                                     INT16(width), // width
00047                                     INT16(height), // height
00048                                     END_OUT_ARGS, // separator
00049                                     &response, // pointer to mem address for return value
00050                                     &fourccc,
00051                                     &renderflags,
00052                                     &xwidth,
00053                                     &ywidth,
00054                                     &size,
00055                                     &videodata, // pointer to mem address for returned frame
00056                                     END_IN_ARGS);
00057
00058     if (return_value == 0) {
00059         return_value = renderBA81(x, y, xwidth, ywidth, size, videodata);
00060     }
00061
00062     return return_value;
00063 }

```

Here is the call graph for this function:



Here is the caller graph for this function:



5.3.2.3 int pixy_render_full_frame (uint16_t x, uint16_t y)

Receives a fullsized frame from pixy and display's it on the display with the topleft corner at (x,y)

Parameters

x	The x-Coordinate of the top left corner
y	The y-Coordinate of the top left corner

Returns

0 on success, otherwise the errorcode from pixy

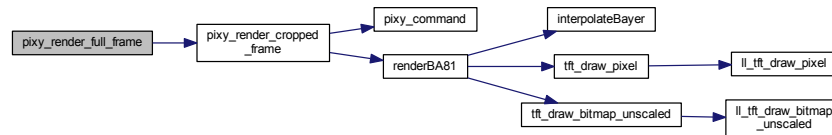
Definition at line 25 of file [pixy_frame.c](#).

```

00026 {
00027     return pixy_render_cropped_frame(x, y, 0, 0, 320, 200);
00028 }

```

Here is the call graph for this function:



Here is the caller graph for this function:



5.3.2.4 int pixy_save_cropped_frame (FILE_HANDLE * handle, uint16_t xoffset, uint16_t yoffset, uint16_t width, uint16_t height)

Receives a cropped frame from pixy and saves it to the given file in the 24bit (b,g,a) format.

Parameters

<i>handle</i>	The file to write the data to. The file must be open and it should be seeked to the right position.
<i>xoffset</i>	The x-Coordinate on the pixy image from where on you want the frame data
<i>yoffset</i>	The y-Coordinate on the pixy image from where on you want the frame data
<i>width</i>	The width of the image recorded from pixy
<i>height</i>	The height of the image recorded from pixy

Returns

0 on success, otherwise the errorcode from pixy

Definition at line 70 of file [pixy_frame.c](#).

```

00071 {
00072     uint8_t* videodata;
00073     int32_t response;
00074     int32_t fourccc;
00075     int8_t renderflags;
00076     uint16_t xwidth;
00077     uint16_t ywidth;
00078     uint32_t size;
00079
00080     int return_value = pixy_command("cam_getFrame", // String id for remote procedure
00081                                     INT8(0x21), // mode
00082                                     INT16(xoffset), // xoffset
00083                                     INT16(yoffset), // yoffset

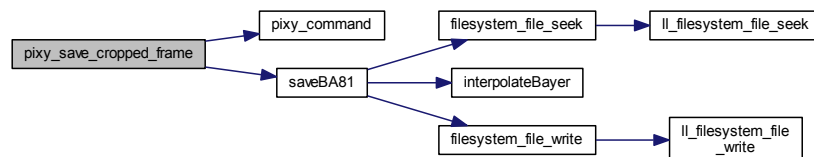
```

```

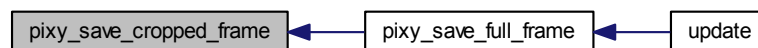
00085             INT16(width),           // width
00086             INT16(height),          // height
00087             END_OUT_ARGS,           // separator
00088             &response,               // pointer to mem address for return value
00089             &fourccc,
00090             &renderflags,
00091             &xwidth,
00092             &ywidth,
00093             &size,
00094             &videodata,             // pointer to mem address for returned frame
00095             END_IN_ARGS);
00096
00097     if (return_value == 0) {
00098         return_value = saveBA81(handle, xwidth, ywidth, size, videodata);
00099     }
00100
00101     return return_value;
00102 }

```

Here is the call graph for this function:



Here is the caller graph for this function:



5.3.2.5 int pixy_save_full_frame (FILE_HANDLE * handle)

Receives a fullsized frame from pixy and saves it to the given file in the 24bit (b,g,a) format. Use this method to write the bitmap-data part of a windows bitmap (.bmp). This method will neither open nor close the passed file.

Parameters

<i>handle</i>	The file to write the data to. The file must be open and it should be seeked to the right position.
---------------	---

Returns

0 on success, otherwise the errorcode from pixy

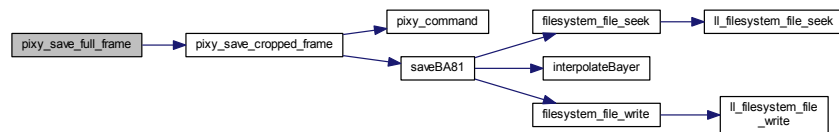
Definition at line 65 of file [pixy_frame.c](#).

```

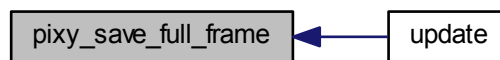
00066 {
00067     return pixy_save_cropped_frame(handle, 0, 0, 320, 200);
00068 }

```


Here is the call graph for this function:

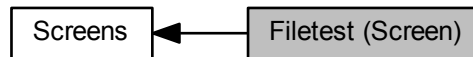


Here is the caller graph for this function:



5.4 Filetest (Screen)

Collaboration diagram for Filetest (Screen):



Functions

- [SCREEN_STRUCT * get_screen_filetest \(\)](#)

5.4.1 Detailed Description

The File-Test Screen tests the filesystem module. It read/writes from/to files and shows a bitmap

5.4.2 Function Documentation

5.4.2.1 SCREEN_STRUCT* get_screen_filetest ()

Returns a pointer to the filetest screen

See also

[gui_screen_navigate](#)

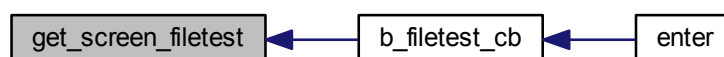
Returns

Definition at line 141 of file [screen_filetest.c](#).

```

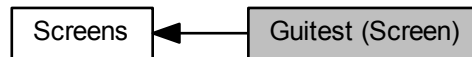
00142 {
00143     return &screen;
00144 }
```

Here is the caller graph for this function:



5.5 Guitest (Screen)

Collaboration diagram for Guitest (Screen):



Functions

- [SCREEN_STRUCT * get_screen_guitest \(\)](#)

5.5.1 Detailed Description

The Gui-Test Screen tests the gui and the tft module.

5.5.2 Function Documentation

5.5.2.1 SCREEN_STRUCT* get_screen_guitest ()

Returns a pointer to the guitest screen

See also

[gui_screen_navigate](#)

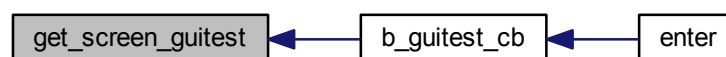
Returns

Definition at line 161 of file [screen_guitest.c](#).

```

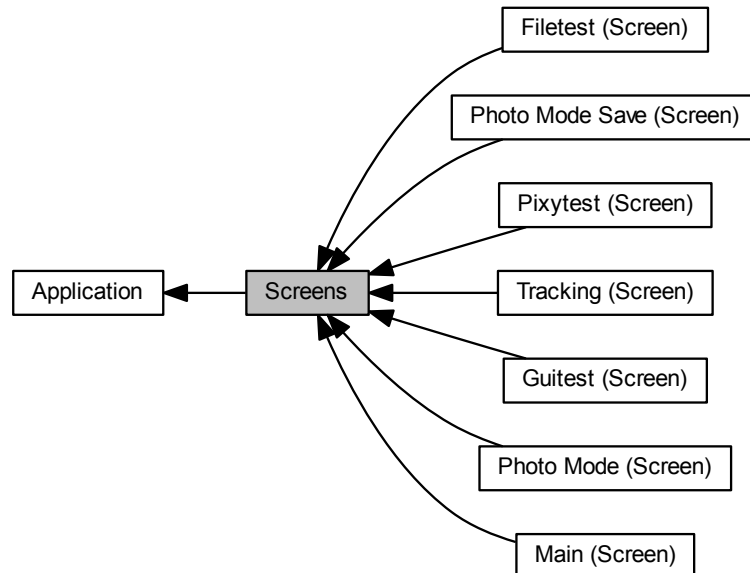
00162 {
00163     return &screen;
00164 }
  
```

Here is the caller graph for this function:



5.6 Screens

Collaboration diagram for Screens:



Modules

- [Filetest \(Screen\)](#)
- [Guittest \(Screen\)](#)
- [Main \(Screen\)](#)
- [Photo Mode \(Screen\)](#)
- [Photo Mode Save \(Screen\)](#)
- [Pixytest \(Screen\)](#)
- [Tracking \(Screen\)](#)

5.6.1 Detailed Description

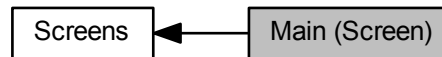
The Screens of the application.

See also

[Screen](#)

5.7 Main (Screen)

Collaboration diagram for Main (Screen):



Functions

- [SCREEN_STRUCT * get_screen_main \(\)](#)

5.7.1 Detailed Description

The Main Screen is the start-screen for the application

5.7.2 Function Documentation

5.7.2.1 [SCREEN_STRUCT* get_screen_main \(\)](#)

Returns a pointer to the main screen

See also

[gui_screen_navigate](#)

Returns

Definition at line [196](#) of file [screen_main.c](#).

```
00197 {  
00198     return &screen;  
00199 }
```

Here is the caller graph for this function:



5.8 Photo Mode (Screen)

Collaboration diagram for Photo Mode (Screen):



Functions

- [SCREEN_STRUCT * get_screen_photomode \(\)](#)

5.8.1 Detailed Description

The Photo Mode Screen allows taking snapshots of the current pixy cam feed

5.8.2 Function Documentation

5.8.2.1 SCREEN_STRUCT* get_screen_photomode ()

Returns a pointer to the photomode screen

See also

[gui_screen_navigate](#)

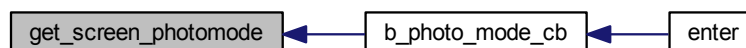
Returns

Definition at line 205 of file [screen_photomode.c](#).

```

00206 {
00207     return &screen;
00208 }
  
```

Here is the caller graph for this function:



5.9 Photo Mode Save (Screen)

Collaboration diagram for Photo Mode Save (Screen):



Functions

- [SCREEN_STRUCT * get_screen_photomodesave \(\)](#)

5.9.1 Detailed Description

The Photo Mode Save Screen helps the user saving a file to the filesystem

5.9.2 Function Documentation

5.9.2.1 [SCREEN_STRUCT*](#) [get_screen_photomodesave \(\)](#)

Returns a pointer to the photomode save screen

See also

[gui_screen_navigate](#)

Returns

Definition at line 338 of file [screen_photomode_save.c](#).

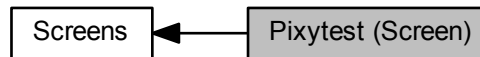
```
00339 {  
00340     return &screen;  
00341 }
```

Here is the caller graph for this function:



5.10 Pixytest (Screen)

Collaboration diagram for Pixytest (Screen):



Functions

- [SCREEN_STRUCT * get_screen_pixytest \(\)](#)

5.10.1 Detailed Description

The Pixy-Test Screen tests the pixy module.

5.10.2 Function Documentation

5.10.2.1 SCREEN_STRUCT* get_screen_pixytest ()

Returns a pointer to the pixytest screen

See also

[gui_screen_navigate](#)

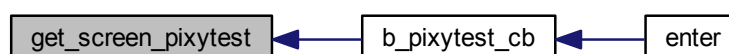
Returns

Definition at line 374 of file [screen_pixytest.c](#).

```

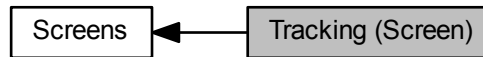
00375 {
00376     return &screen;
00377 }
```

Here is the caller graph for this function:



5.11 Tracking (Screen)

Collaboration diagram for Tracking (Screen):



Enumerations

- enum [Tracking_Implementation](#) { [OUR_TRACKING](#), [REFERENCE_TRACKING](#) }

Functions

- void [tracking_set_mode](#) (enum [Tracking_Implementation](#) impl)
- [SCREEN_STRUCT](#) * [get_screen_tracking](#) ()

5.11.1 Detailed Description

The Tracking-Screen shows the object-tracking and allows some configuration

5.11.2 Enumeration Type Documentation

5.11.2.1 enum [Tracking_Implementation](#)

Enum which contains the available tracking implementations

Enumerator

[OUR_TRACKING](#) Our own tracking PID implementation.

[REFERENCE_TRACKING](#) Pixy's internal tracking implementation.

Definition at line [31](#) of file [screen_tracking.h](#).

```

00031                                     {
00032     OUR\_TRACKING,
00033     REFERENCE\_TRACKING
00034 };
  
```

5.11.3 Function Documentation

5.11.3.1 [SCREEN_STRUCT](#)* [get_screen_tracking](#) ()

Returns a pointer to the tracking screen

See also

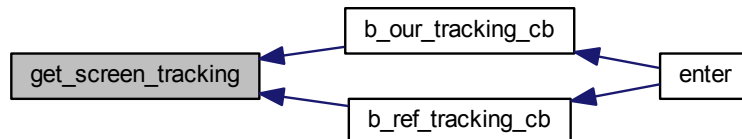
[gui_screen_navigate](#)

Returns

Definition at line 418 of file [screen_tracking.c](#).

```
00419 {
00420     return &screen;
00421 }
```

Here is the caller graph for this function:



5.11.3.2 void tracking_set_mode (enum Tracking_Implementation impl)

Sets the current Mode/Tracking Implementation. Call this before using the screen obtained by [get_screen_tracking\(\)](#)

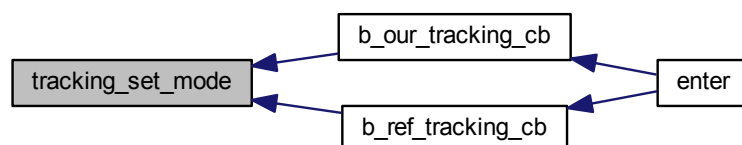
Parameters

<i>impl</i>	The new mode
-------------	--------------

Definition at line 210 of file [screen_tracking.c](#).

```
00211 {
00212     //Depending on the enum value let tracking_current point to a different setting/callback structure
00213     switch (impl) {
00214     case OUR_TRACKING:
00215         tracking_current = &tracking_our;
00216         break;
00217     case REFERENCE_TRACKING:
00218         tracking_current = &tracking_reference;
00219         break;
00220     default:
00221         tracking_current = NULL;
00222         break;
00223     }
00224 }
00225 }
00226 }
```

Here is the caller graph for this function:



5.12 Filesystem

Data Structures

- struct [FILE_DATE_STRUCT](#)
- struct [FILE_TIME_STRUCT](#)
- struct [FILE_STRUCT](#)
- struct [DIRECTORY_STRUCT](#)
- struct [FILE_HANDLE](#)

Enumerations

- enum [FILE_ATTRIBUTES](#) {
[F_RDO](#) = 0x01, [F_HID](#) = 0x02, [F_SYS](#) = 0x04, [F_DIR](#) = 0x10,
[F_ARC](#) = 0x20 }
- enum [FILE_STATUS](#) {
[F_OK](#), [F_EOF](#), [F_EACCESS](#), [F_INVALIDPARAM](#),
[F_DISKERROR](#) }

Functions

- bool [filesystem_init](#) ()
- [DIRECTORY_STRUCT](#) * [filesystem_dir_open](#) (const char *path)
- void [filesystem_dir_close](#) ([DIRECTORY_STRUCT](#) *dir)
- [FILE_HANDLE](#) * [filesystem_file_open](#) (const char *filename)
- void [filesystem_file_close](#) ([FILE_HANDLE](#) *handle)
- [FILE_STATUS](#) [filesystem_file_seek](#) ([FILE_HANDLE](#) *handle, uint32_t offset)
- [FILE_STATUS](#) [filesystem_file_read](#) ([FILE_HANDLE](#) *handle, uint8_t *buf, uint32_t size)
- [FILE_STATUS](#) [filesystem_file_write](#) ([FILE_HANDLE](#) *handle, uint8_t *buf, uint32_t size)

5.12.1 Detailed Description

The Filesystem Module provides access to files and directories of a the native filesystem.

5.12.2 Enumeration Type Documentation

5.12.2.1 enum [FILE_ATTRIBUTES](#)

File Attributes used by implementation See http://en.wikipedia.org/wiki/Design_of_the_FAT_file_system#attributes for detailed description

Enumerator

- [F_RDO](#)** File is readonly. You cannot write to it.
- [F_HID](#)** File is hidden.
- [F_SYS](#)** File is a system file.
- [F_DIR](#)** It's a directory and not a file.
- [F_ARC](#)** File has the archive flag set (probably unused)

Definition at line 32 of file [filesystem.h](#).

```

00032     {
00033         F_RDO = 0x01,
00034         F_HID = 0x02,
00035         F_SYS = 0x04,
00036         F_DIR = 0x10,
00037         F_ARC = 0x20
00038     } FILE_ATTRIBUTES;

```

5.12.2.2 enum FILE_STATUS

Enum to represent the success or error-code of the filesystem_file_* functions

Enumerator

- F_OK** Everything ok.
- F_EOF** The write/read operation tried to write/read past the end of the file. This is not a fatal error.
- F_EACCESS** The file can not be read/written due to access problems. This is a fatal error.
- F_INVALIDPARAM** You passed invalid parameters to the function.
- F_DISKERROR** A lowlevel disk-error occurred. This is a fatal error.

Definition at line 90 of file [filesystem.h](#).

```

00090     {
00091         F_OK,
00092         F_EOF,
00093         F_EACCESS,
00094         F_INVALIDPARAM,
00095         F_DISKERROR
00096     } FILE_STATUS;

```

5.12.3 Function Documentation

5.12.3.1 void filesystem_dir_close (DIRECTORY_STRUCT * dir)

Closes a previously opened directory. Free's all allocated resources.

Parameters

<i>dir</i>	A Pointer to a DIRECTORY_STRUCT obtained by filesystem_dir_open() .
------------	---

Definition at line 28 of file [filesystem.c](#).

```

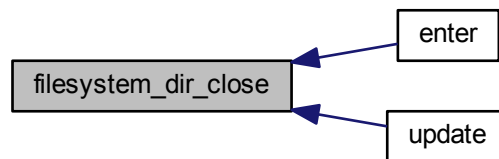
00029 {
00030     ll_filesystem_dir_close(dir);
00031 }

```

Here is the call graph for this function:



Here is the caller graph for this function:



5.12.3.2 `DIRECTORY_STRUCT*` `filesystem_dir_open` (`const char *` *path*)

Opens a directory and returns a structure which contains all files/subdirectories.

See also

[filesystem_dir_close\(\)](#)

Parameters

<i>path</i>	The absolute path to the directory to open/read
-------------	---

Returns

A Pointer to an initialized [DIRECTORY_STRUCT](#) on success, NULL on error

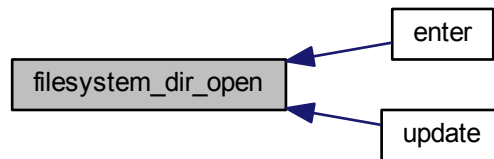
Definition at line 23 of file [filesystem.c](#).

```
00024 {  
00025     return ll_filesystem_dir_open(path);  
00026 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.12.3.3 void filesystem_file_close (FILE_HANDLE * handle)

Closes a file.

Parameters

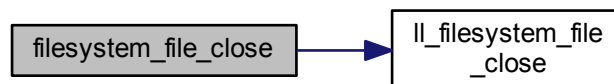
<i>handle</i>	The FILE_HANDLE obtained by filesystem_file_open()
---------------	--

Definition at line 38 of file [filesystem.c](#).

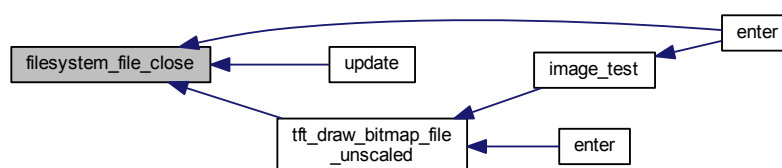
```

00039 {
00040     ll_filesystem_file_close(handle);
00041 }
  
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.12.3.4 FILE_HANDLE* filesystem_file_open (const char * filename)

Opens a file for read/writing.

Note

Depending on the implementation you may only open one file at a time

Parameters

<i>filename</i>	The absolute file path
-----------------	------------------------

Returns

A Pointer to a [FILE_HANDLE](#) on success, NULL on error.

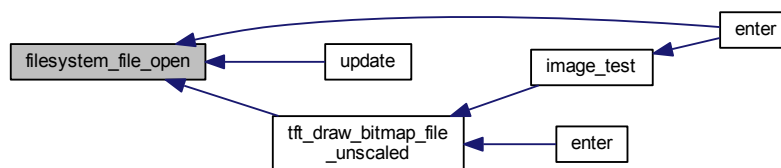
Definition at line 33 of file [filesystem.c](#).

```
00034 {
00035     return ll_filesystem_file_open(filename);
00036 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.12.3.5 FILE_STATUS filesystem_file_read (FILE_HANDLE * handle, uint8_t * buf, uint32_t size)

Reads some bytes from an open file.

Parameters

<i>handle</i>	The FILE_HANDLE obtained by filesystem_file_open()
<i>buf</i>	The Buffer to write the bytes to
<i>size</i>	The number of bytes to read

Returns

[F_OK](#) on success, [F_EOF](#) if less than `size` bytes could be read, an error Code otherwise.

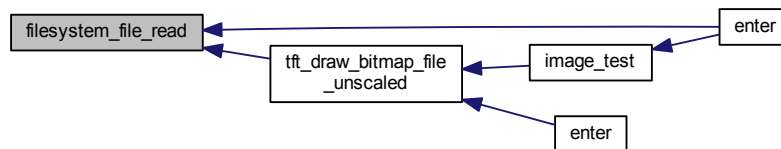
Definition at line 48 of file [filesystem.c](#).

```
00049 {
00050     return ll_filesystem_file_read(handle, buf, size);
00051 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.12.3.6 FILE_STATUS filesystem_file_seek (FILE_HANDLE * handle, uint32_t offset)

Set's the read/write position to a new position

Parameters

<i>handle</i>	The FILE_HANDLE obtained by filesystem_file_open()
<i>offset</i>	The new read/write position in bytes (absolute).

Returns

[F_OK](#) on success, an error Code otherwise.

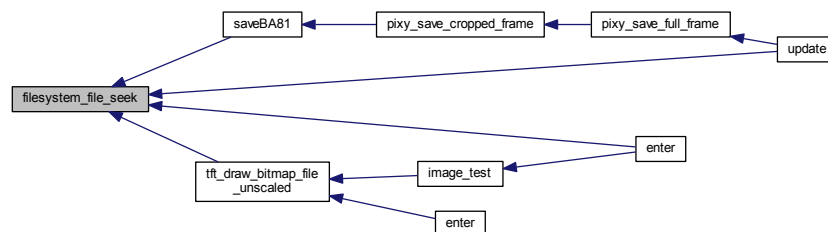
Definition at line 43 of file [filesystem.c](#).

```
00044 {
00045     return ll_filesystem_file_seek(handle, offset);
00046 }
```


Here is the call graph for this function:



Here is the caller graph for this function:



5.12.3.7 FILE_STATUS filesystem_file_write (FILE_HANDLE * handle, uint8_t * buf, uint32_t size)

Writes some bytes to a open file.

Note

Depending on the implementation the file may not be shrinked or expanded.

Parameters

<i>handle</i>	The FILE_HANDLE obtained by filesystem_file_open()
<i>buf</i>	The Buffer to take the bytes from
<i>size</i>	The number of bytes to write

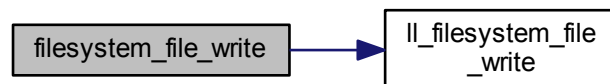
Returns

`F_OK` on success, `F_EOF` if less than `size` bytes could be written, an error Code otherwise.

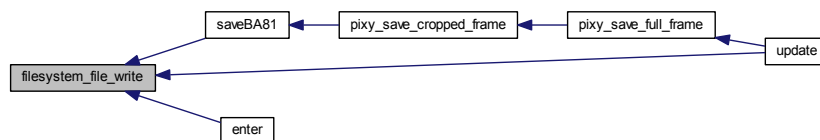
Definition at line 53 of file `filesystem.c`.

```
00054 {
00055     return ll_filesystem_file_write(handle, buf, size);
00056 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.12.3.8 bool filesystem_init ()

Initializes the filesystem. Call this method before using any `filesystem_*` functions

Returns

true on success

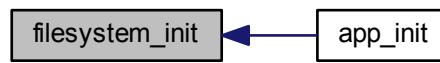
Definition at line 18 of file `filesystem.c`.

```
00019 {
00020     return ll_filesystem_init();
00021 }
```

Here is the call graph for this function:

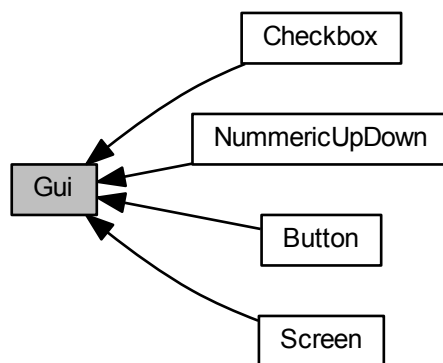


Here is the caller graph for this function:



5.13 Gui

Collaboration diagram for Gui:



Modules

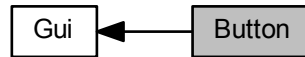
- [Button](#)
- [Checkbox](#)
- [NummericUpDown](#)
- [Screen](#)

5.13.1 Detailed Description

The Gui Module

5.14 Button

Collaboration diagram for Button:



Data Structures

- struct [BUTTON_STRUCT](#)

Macros

- `#define` [AUTO](#) 0

Use this value instead of `x2`, `y2` in the [BUTTON_STRUCT](#) to autocalculate the button width/height.

Typedefs

- `typedef void(*` [BUTTON_CALLBACK](#) `)(void *button)`

Functions

- `bool` [gui_button_add](#) ([BUTTON_STRUCT](#) *button)
- `void` [gui_button_remove](#) ([BUTTON_STRUCT](#) *button)
- `void` [gui_button_redraw](#) ([BUTTON_STRUCT](#) *button)

5.14.1 Detailed Description

The Button Gui-Element is a clickable, rectangular box with a label inside. When it is pressed and released you will be notified via the provided callback.

5.14.2 Macro Definition Documentation

5.14.2.1 `#define` [AUTO](#) 0

Use this value instead of `x2`, `y2` in the [BUTTON_STRUCT](#) to autocalculate the button width/height.

Definition at line [65](#) of file [button.h](#).

5.14.3 Typedef Documentation

5.14.3.1 `typedef void(*` [BUTTON_CALLBACK](#) `)(void *button)`

Prototype for Event Listeners (called when the button is pressed)

Note

You should NOT execute long running things in this callback nor should you update the gui. But you can call [gui_screen_navigate\(\)](#) for instance.

Parameters

<i>button</i>	The pointer to the BUTTON_STRUCT where to corresponding Button was pressed
---------------	--

Definition at line 49 of file [button.h](#).

5.14.4 Function Documentation**5.14.4.1 bool gui_button_add ([BUTTON_STRUCT](#) * *button*)**

Adds a button. Your Callback will be called from now on, if the button was pressed

Parameters

<i>button</i>	A Pointer to the preinitialized BUTTON_STRUCT
---------------	---

Returns

true on success

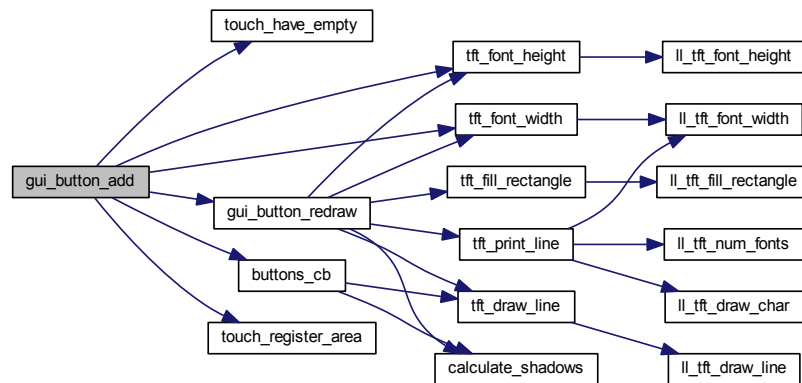
Definition at line 133 of file [button.c](#).

```

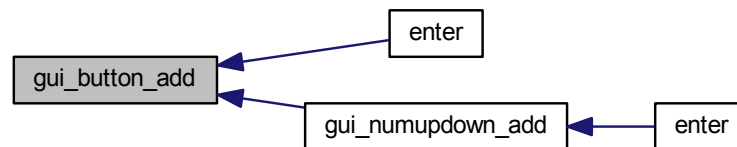
00134 {
00135     if (touch_have_empty(1)) { //Check if the touch module can handle one additional area
00136         //Calculate width and height of the button text
00137         unsigned int strwidth = tft_font_width(button->font) * strlen(button->
text);
00138         unsigned char strheight = tft_font_height(button->font);
00139
00140         button->base.hookedActions = PEN_DOWN; //At first we are interested in
PEN_DOWN events
00141         button->base.callback = buttons_cb; //Use our own callback for the touch area
events
00142
00143         if (button->base.x2 == AUTO) { //The user wants us to calculate the button width
automatically
00144             //Use string width + half of a character width as button width
00145             button->base.x2 = button->base.x1 - 1 + strwidth + (
tft_font_width(button->font) / 2);
00146         } else if ((button->base.x2 - button->base.x1 + 1) < (strwidth + 2)) { //the provided
width is too small to fit the entire text
00147             return false; //report error
00148         }
00149
00150         if (button->base.y2 == AUTO) { //The user wants us to calculate the button height
automatically
00151             //Use one and a half character heights as button height
00152             button->base.y2 = button->base.y1 - 1 + strheight + (strheight / 2);
00153         } else if ((button->base.y2 - button->base.y1 + 1) < (strheight + 2)) { //the provided
height is too small to fit the text
00154             return false;
00155         }
00156
00157         gui_button_redraw(button); //call the redraw method, which will take care of
drawing the entire button
00158         return touch_register_area(&button->base); //Register the touch area and
receive events for this button, from now on
00159     }
00160
00161     return false; //no more touch areas left
00162 }

```

Here is the call graph for this function:



Here is the caller graph for this function:



5.14.4.2 void gui_button_redraw (BUTTON_STRUCT * button)

Redraws the button. Call this method if you have to redraw the entire screen or if you want to draw a button on top of an image.

Parameters

<i>button</i>	A Pointer to the BUTTON_STRUCT
---------------	--

Definition at line 164 of file [button.c](#).

```

00165 {
00166     //Calculate text dimensions and shadow colors
00167     unsigned int strwidth = tft_font_width(button->font) * strlen(button->
text);
00168     unsigned char strheight = tft_font_height(button->font);
00169     uint16_t c_light, c_dark;
00170     calculate_shadows(button->bgcolor, &c_light, &c_dark);
00171
00172     //Draw the background and the 4 lines (shadow colors)
00173     tft_fill_rectangle(button->base.x1 + 1, button->base.
y1 + 1, button->base.x2 - 1, button->base.y2 - 1, button->bgcolor);
00174     tft_draw_line(button->base.x1 + 1, button->base.y1, button->
base.x2 - 1, button->base.y1, c_light); //North
00175     tft_draw_line(button->base.x1, button->base.y1 + 1, button->
base.x1, button->base.y2 - 1, c_light); //West
00176     tft_draw_line(button->base.x1 + 1, button->base.y2, button->
base.x2 - 1, button->base.y2, c_dark); //South

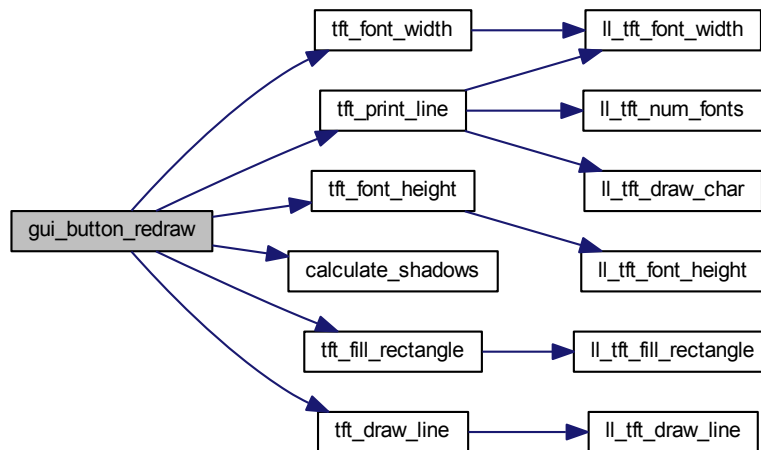
```

```

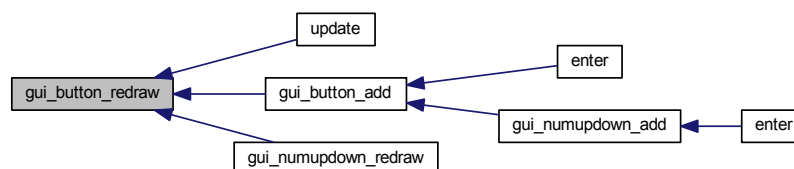
00177     tft_draw_line(button->base.x2, button->base.y1 + 1, button->
base.x2, button->base.y2 - 1, c_dark); //East
00178
00179     //Draw the text
00180     tft_print_line(button->base.x1 + (button->base.x2 - button->
base.x1 + 1 - strwidth) / 2, button->base.y1 + (button->base.y2 - button->
base.y1 + 1 - strheight) / 2, button->txtcolor, button->bghcolor, button->
font, button->text);
00181
00182 }

```

Here is the call graph for this function:



Here is the caller graph for this function:



5.14.4.3 void gui_button_remove (BUTTON_STRUCT * button)

Removes the button. You will no longer receive events for this button. This function will not overdraw the region where the button was located.

Parameters

<i>button</i>	A Pointer to the BUTTON_STRUCT
---------------	--

Definition at line 184 of file [button.c](#).

```

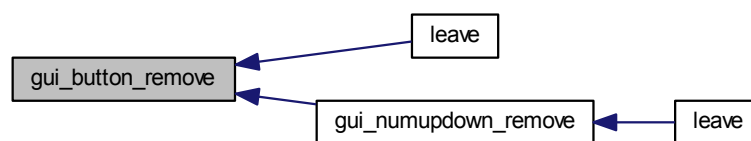
00185 {
00186     //We only need to unregister the touch area, as we have not allocated anything else
00187     touch_unregister_area((TOUCH_AREA_STRUCT*)button);
00188 }

```


Here is the call graph for this function:



Here is the caller graph for this function:



5.15 Checkbox

Collaboration diagram for Checkbox:



Data Structures

- struct [CHECKBOX_STRUCT](#)

Macros

- #define [CHECKBOX_WIN_FG_COLOR](#) RGB(32,161,34)

Typedefs

- typedef void(* [CHECKBOX_CALLBACK](#)) (void *checkbox, bool checked)

Functions

- bool [gui_checkbox_add](#) ([CHECKBOX_STRUCT](#) *checkbox)
- void [gui_checkbox_remove](#) ([CHECKBOX_STRUCT](#) *checkbox)
- void [gui_checkbox_update](#) ([CHECKBOX_STRUCT](#) *checkbox)
- void [gui_checkbox_redraw](#) ([CHECKBOX_STRUCT](#) *checkbox)

5.15.1 Detailed Description

The Checkbox Gui-Element is a clickable, rectangular box with an optional tickmark inside of it. When the checkbox is pressed and released it's tick state changes and you will be notified via the provided callback.

5.15.2 Macro Definition Documentation

5.15.2.1 #define CHECKBOX_WIN_FG_COLOR RGB(32,161,34)

Definition at line 82 of file [checkbox.h](#).

5.15.3 Typedef Documentation

5.15.3.1 typedef void(* CHECKBOX_CALLBACK) (void *checkbox, bool checked)

Prototype for Event Listeners (called when the checkbox state has changed)

Note

You should NOT execute long running things in this callback nor should you update the gui. But you can call `gui_screen_navigate()` for instance.

Parameters

<i>checkbox</i>	The pointer to the <code>CHECKBOX_STRUCT</code> where to corresponding Checkbox has changed the state
<i>checked</i>	A boolean which indicates whether the checkbox is now checked or not.

Definition at line 45 of file `checkbox.h`.

5.15.4 Function Documentation**5.15.4.1 bool gui_checkbox_add (CHECKBOX_STRUCT * checkbox)**

Adds a checkbox. Your Callback will be called from now on, if the checkbox changes state

Parameters

<i>checkbox</i>	A Pointer to the preinitialized <code>CHECKBOX_STRUCT</code>
-----------------	--

Returns

true on success

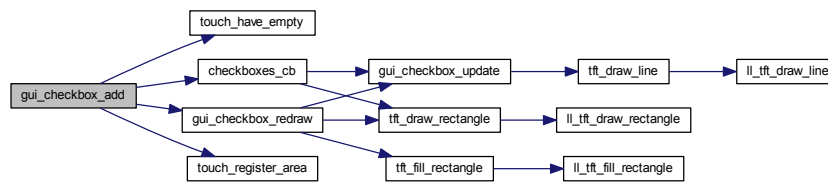
Definition at line 70 of file `checkbox.c`.

```

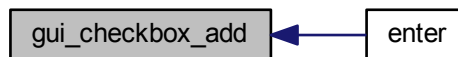
00071 {
00072     if (touch_have_empty(1)) { //Check if the touch module can handle one additional area
00073         unsigned char size = 0;
00074         checkbox->base.hookedActions = PEN_DOWN; //At first we are interested in
PEN_DOWN events
00075         checkbox->base.callback = checkboxes_cb; //Use our own callback for the
touch area events
00076
00077         //Check the size of the checkbox
00078         if (checkbox->base.x2 > checkbox->base.x1) {
00079             size = checkbox->base.x2 - checkbox->base.x1; //use width a as size
00080         }
00081
00082         if (checkbox->base.y2 > checkbox->base.y1) {
00083             if ((checkbox->base.y2 - checkbox->base.y1) > size) { //height is larger than size
00084                 size = checkbox->base.y2 - checkbox->base.y1; //use height as size
00085             }
00086         }
00087
00088         if (size == 0) { //no size found (maybe swap x2 and x1 or y2 and y1 ?)
00089             return false; //signal error
00090         }
00091
00092         if ((size & 0x01)) { //the size is an odd number
00093             size++; //make size an even number
00094         }
00095
00096         //Correct x2,y2 so that the checkbox is quadratic
00097         checkbox->base.x2 = checkbox->base.x1 + size;
00098         checkbox->base.y2 = checkbox->base.y1 + size;
00099
00100         gui_checkbox_redraw(checkbox); //Call redraw method, which will take care of the
drawing of the entire checkbox
00101
00102         return touch_register_area(&checkbox->base); //Register the touch area and
receive events for this checkbox, from now on
00103     }
00104
00105     return false; //no more touch areas left
00106 }

```

Here is the call graph for this function:



Here is the caller graph for this function:



5.15.4.2 void gui_checkbox_redraw (CHECKBOX_STRUCT * checkbox)

Redraws the checkbox. Call this method if you have to redraw the entire screen or if you want to draw a checkbox on top of an image.

Parameters

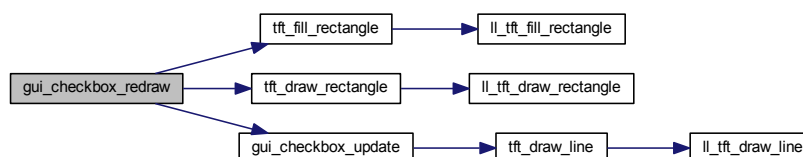
<i>checkbox</i>	A Pointer to the CHECKBOX_STRUCT
-----------------	--

Definition at line 108 of file [checkbox.c](#).

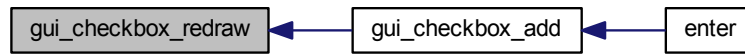
```

00109 {
00110     //Draw background and border
00111     tft_fill_rectangle(checkbox->base.x1 + 1, checkbox->
base.y1 + 1, checkbox->base.x2 - 1, checkbox->base.y2 - 1,
BACKGROUND_COLOR);
00112     tft_draw_rectangle(checkbox->base.x1, checkbox->base.
y1, checkbox->base.x2, checkbox->base.y2, BORDER_COLOR);
00113
00114     if (checkbox->checked) { //checkbox is currently checked
00115         gui_checkbox_update(checkbox); //Call update method which will draw the tickmark
00116     }
00117 }
  
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.15.4.3 void gui_checkbox_remove (CHECKBOX_STRUCT * checkbox)

Removes the checkbox. You will no longer receive events for this checkbox. This function will not overdraw the region where the checkbox was located.

Parameters

<i>checkbox</i>	A Pointer to the CHECKBOX_STRUCT
-----------------	--

Definition at line 119 of file [checkbox.c](#).

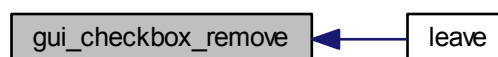
```

00120 {
00121     //We only need to unregister the touch area, as we have not allocated anything else
00122     touch_unregister_area((TOUCH_AREA_STRUCT*)checkbox);
00123 }
  
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.15.4.4 void gui_checkbox_update (CHECKBOX_STRUCT * checkbox)

Updates the checkbox. Call this function when you change the state of the checkbox through code.

Parameters

<i>checkbox</i>	A Pointer to the CHECKBOX_STRUCT
-----------------	--

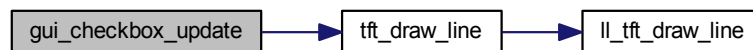
Definition at line 125 of file [checkbox.c](#).

```

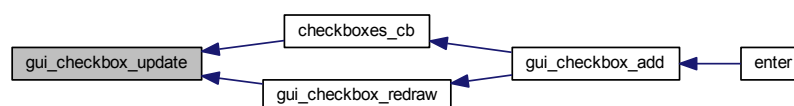
00126 {
00127     unsigned int c = (checkbox->checked) ? checkbox->fgcolor :
    BACKGROUND_COLOR; //color to use for the tickmark
00128
00129     //helper points inside the checkbox
00130     unsigned int xcent = checkbox->base.x1 + (checkbox->base.x2 - checkbox->
    base.x1) * 6 / 14;
00131     unsigned int yleft = checkbox->base.y2 - (xcent - checkbox->base.
    x1) - 1;
00132     unsigned int yright = checkbox->base.y2 - (checkbox->base.x2 - xcent) - 1;
00133     unsigned int ybot = checkbox->base.y2 - 4;
00134
00135     //Draw tickmark as a 3pixel wide line
00136     tft_draw_line(checkbox->base.x1 + 3, yleft - 1, xcent, ybot - 1, c);
00137     tft_draw_line(checkbox->base.x1 + 3, yleft, xcent, ybot, c);
00138     tft_draw_line(checkbox->base.x1 + 3, yleft + 1, xcent, ybot + 1, c);
00139     xcent++;
00140     ybot--;
00141     tft_draw_line(xcent, ybot - 1, checkbox->base.x2 - 3, yright - 1, c);
00142     tft_draw_line(xcent, ybot, checkbox->base.x2 - 3, yright + 0, c);
00143     tft_draw_line(xcent, ybot + 1, checkbox->base.x2 - 3, yright + 1, c);
00144 }

```

Here is the call graph for this function:

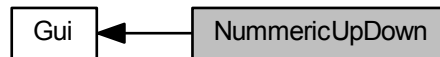


Here is the caller graph for this function:



5.16 NummericUpDown

Collaboration diagram for NummericUpDown:



Data Structures

- struct [NUMUPDOWN_STRUCT](#)

Typedefs

- typedef void(* [NUMUPDOWN_CALLBACK](#)) (void *numupdown, int16_t value)

Functions

- bool [gui_numupdown_add](#) ([NUMUPDOWN_STRUCT](#) *numupdown)
- void [gui_numupdown_remove](#) ([NUMUPDOWN_STRUCT](#) *numupdown)
- void [gui_numupdown_update](#) ([NUMUPDOWN_STRUCT](#) *numupdown)
- void [gui_numupdown_redraw](#) ([NUMUPDOWN_STRUCT](#) *numupdown)

5.16.1 Detailed Description

The NummericUpDown Gui Element

5.16.2 Typedef Documentation

5.16.2.1 typedef void(* [NUMUPDOWN_CALLBACK](#)) (void *numupdown, int16_t value)

Prototype for Event Listeners (called when the NummericUpDown value has changed)

Note

You should NOT execute long running things in this callback nor should you update the gui. But you can call [gui_screen_navigate\(\)](#) for instance.

Parameters

<i>numupdown</i>	The pointer to the NUMUPDOWN_STRUCT where to corresponding NummericUpDown has changed it's value
------------------	--

<i>value</i>	The new value of the NumericUpDown
--------------	------------------------------------

Definition at line 43 of file [numupdown.h](#).

5.16.3 Function Documentation

5.16.3.1 bool gui_numupdown_add (NUMUPDOWN_STRUCT * numupdown)

Adds a NumericUpDown. Your Callback will be called from now on, if the numupdown's value changes

Parameters

<i>numupdown</i>	A Pointer to the preinitialized NUMUPDOWN_STRUCT
------------------	--

Returns

true on success

Definition at line 80 of file [numupdown.c](#).

```

00081 {
00082     if (touch_have_empty(2)) { //Check if the touch module can handle two additional areas
00083         if (numupdown->min > numupdown->max) { //min is bigger than max?
00084             return false; //invalid parameter
00085         }
00086
00087         if (numupdown->value < numupdown->min) { //value is smaller than min?
00088             numupdown->value = numupdown->min; //normalize value
00089         } else if (numupdown->value > numupdown->max) { //value is bigger than max?
00090             numupdown->value = numupdown->max; //normalize value
00091         }
00092
00093         uint8_t tw1 = calc_text_width(numupdown->max); //Calculate character width to
render maximum value
00094         uint8_t tw2 = calc_text_width(numupdown->min); //Calculate character width to
render minimum value
00095
00096         if (tw2 > tw1) {
00097             tw1 = tw2; //ensure tw1 contains the larger number of the two
00098         }
00099
00100         uint8_t width = tft_font_width(0) * (tw1 + 1); //Calculate width of the number
area
00101
00102         //Add "minus" button to the left side of the number area
00103         numupdown->buttonDown.base.x1 = numupdown->x;
00104         numupdown->buttonDown.base.y1 = numupdown->y;
00105         numupdown->buttonDown.base.x2 = AUTO;
00106         numupdown->buttonDown.base.y2 = numupdown->y +
tft_font_height(0) * 2;
00107         numupdown->buttonDown.text = "-";
00108         numupdown->buttonDown.font = 0;
00109         numupdown->buttonDown.bgcolor = BASE_COLOR;
00110         numupdown->buttonDown.txtcolor = WHITE;
00111         numupdown->buttonDown.callback = button_down_cb;
00112         gui_button_add(&numupdown->buttonDown);
00113
00114         //Add "plus" button to the right side of the number area
00115         numupdown->buttonUp.base.x1 = numupdown->buttonDown.
base.x2
+ width + 2;
00116         numupdown->buttonUp.base.y1 = numupdown->y;
00117         numupdown->buttonUp.base.x2 = AUTO;
00118         numupdown->buttonUp.base.y2 = numupdown->y +
tft_font_height(0) * 2;
00119         numupdown->buttonUp.text = "+";
00120         numupdown->buttonUp.font = 0;
00121         numupdown->buttonUp.bgcolor = BASE_COLOR;
00122         numupdown->buttonUp.txtcolor = WHITE;
00123         numupdown->buttonUp.callback = button_up_cb;
00124         gui_button_add(&numupdown->buttonUp);
00125
00126         //Draw background and label of the number area
00127         tft_fill_rectangle(numupdown->buttonDown.
base.x2 + 2, numupdown->y, numupdown->buttonDown.base.x2 + width, numupdown->
buttonUp.base.y2, BASE_COLOR);
00128         tft_print_formatted(numupdown->buttonDown.
base.x2 + 2 + tft_font_width(0) / 2, numupdown->y +

```

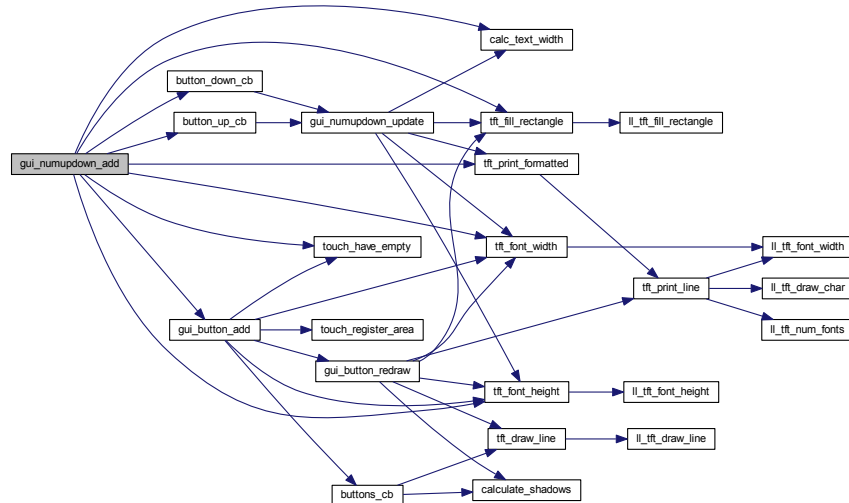


```

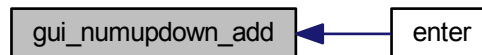
00129     tft_font_height(0) / 2, numupdown->fgcolor, BASE_COLOR, 0, "%*d", tw1,
numupdown->value);
00130     return true;
00131 }
00132
00133     return false; //not enough touch areas left
00134 }

```

Here is the call graph for this function:



Here is the caller graph for this function:



5.16.3.2 void gui_numupdown_redraw (NUMUPDOWN_STRUCT * numupdown)

Redraws the NumericUpDown. Call this method if you have to redraw the entire screen or if you want to draw a numupdown on top of an image.

Parameters

<i>numupdown</i>	A Pointer to the NUMUPDOWN_STRUCT
------------------	---

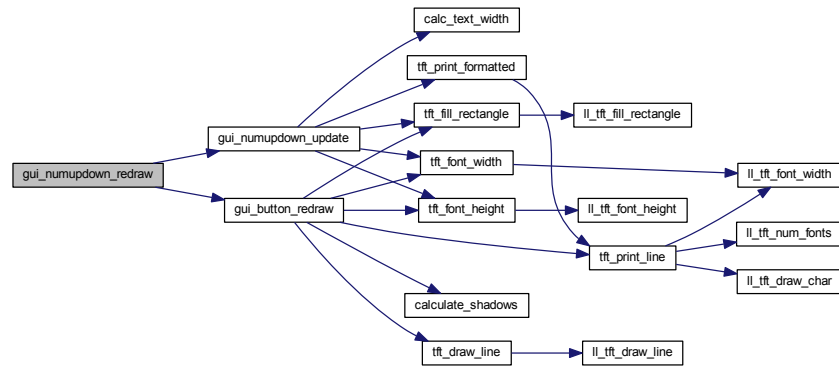
Definition at line 144 of file [numupdown.c](#).

```

00145 {
00146     //redraw the two buttons
00147     gui_button_redraw(&numupdown->buttonUp);
00148     gui_button_redraw(&numupdown->buttonDown);
00149
00150     //call update method which will take care of the number-area rendering
00151     gui_numupdown_update(numupdown);
00152 }

```

Here is the call graph for this function:



5.16.3.3 void gui_numupdown_remove (NUMUPDOWN_STRUCT * numupdown)

Removes the NumericUpDown. You will no longer receive events for this numupdown. This function will not overdraw the region where the numupdown was located.

Parameters

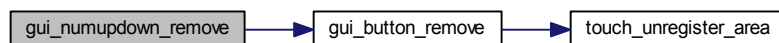
<i>numupdown</i>	A Pointer to the NUMUPDOWN_STRUCT
------------------	---

Definition at line 136 of file [numupdown.c](#).

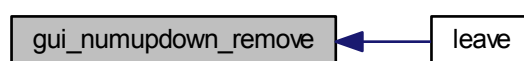
```

00137 {
00138     //remove the two buttons, we have no other allocated resources
00139     gui_button_remove (&numupdown->buttonUp);
00140     gui_button_remove (&numupdown->buttonDown);
00141 }
  
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.16.3.4 void gui_numupdown_update (NUMUPDOWN_STRUCT * *numupdown*)

Updates the NummericUpDown. Call this function when you change the value/min/max of the numupdown through code.

Parameters

<i>numupdown</i>	A Pointer to the NUMUPDOWN_STRUCT
------------------	---

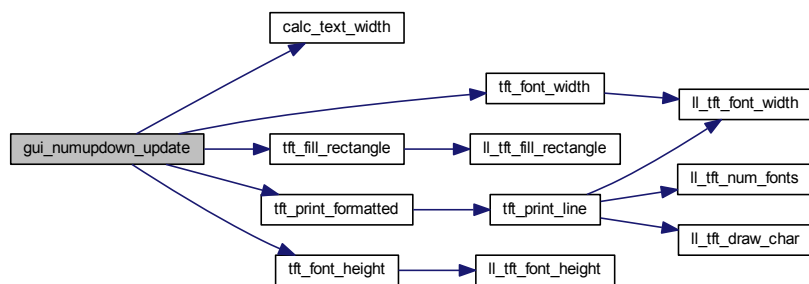
Definition at line 154 of file [numupdown.c](#).

```

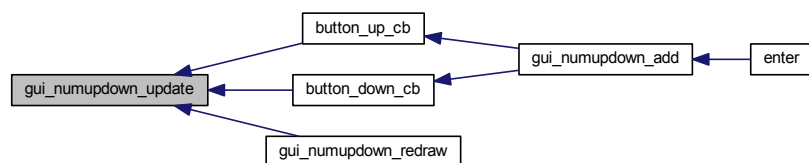
00155 {
00156     //Calculate the number area width again (see above)
00157     uint8_t tw1 = calc_text_width(numupdown->max);
00158     uint8_t tw2 = calc_text_width(numupdown->min);
00159
00160     if (tw2 > tw1) {
00161         tw1 = tw2;
00162     }
00163
00164     uint8_t width = tft_font_width(0) * (tw1 + 1);
00165
00166     //Draw background and label of the number area
00167     tft_fill_rectangle(numupdown->buttonDown.base.x2 + 2, numupdown->y, numupdown->buttonDown.base.x2 + width, numupdown->buttonUp.base.y2, BASE_COLOR);
00168     tft_print_formatted(numupdown->buttonDown.base.x2 + 2 + tft_font_width(0) / 2, numupdown->y + tft_font_height(0) / 2, numupdown->fgcolor, BASE_COLOR, 0, "%*d", tw1, numupdown->value);
00169 }

```

Here is the call graph for this function:

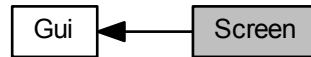


Here is the caller graph for this function:



5.17 Screen

Collaboration diagram for Screen:



Data Structures

- struct [SCREEN_S](#)

Typedefs

- typedef void(* [SCREEN_CALLBACK](#)) (void *[screen](#))
- typedef struct [SCREEN_S](#) [SCREEN_STRUCT](#)

Functions

- bool [gui_screen_navigate](#) ([SCREEN_STRUCT](#) *[screen](#))
- bool [gui_screen_back](#) ()
- [SCREEN_STRUCT](#) * [gui_screen_get_current](#) ()
- void [gui_screen_update](#) ()

5.17.1 Detailed Description

The Screen Submodule provides an api to navigate between different "screens" on the UI. Each screen must provide an enter, update and a leave method; which will be called from this module at the right time. The implemented screens of the application are documented in the [Screens](#) module.

5.17.2 Typedef Documentation

5.17.2.1 typedef void(* [SCREEN_CALLBACK](#)) (void *[screen](#))

Prototype for Event Listeners (called when the screen is entered, left or should be updated)

Parameters

<i>screen</i>	The pointer to the SCREEN_STRUCT where the event occurred
---------------	---

Definition at line 47 of file [screen.h](#).

5.17.2.2 typedef struct [SCREEN_S](#) [SCREEN_STRUCT](#)

Structure to configure the Screen

5.17.3 Function Documentation

5.17.3.1 bool gui_screen_back()

Navigate one screen back as soon as the app enters the main loop again. It's safe to call this method from an interrupt

Returns

true on success

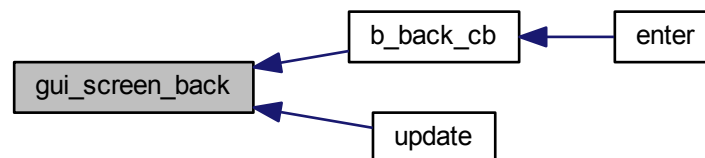
Definition at line 85 of file [screen.c](#).

```

00086 {
00087     if (screen_list == NULL) { //the list head is empty, nothing to go back to
00088         return false;
00089     }
00090
00091     SCREEN_STRUCT* current = screen_list;
00092     SCREEN_STRUCT* last = NULL;
00093
00094     //Find second last element in list
00095     while (current->next != NULL) {
00096         last = current;
00097         current = current->next;
00098     }
00099
00100     if (last == NULL) {
00101         return false;    //There's only a single screen, there's no going back here
00102     }
00103
00104     if (current != screen_current) {
00105         return false;    //The last entry in the list is not the current screen. List corrupted?
00106     }
00107
00108     screen_goto = last; // "send message" to main loop, to switch the screen
00109     return true;
00110 }

```

Here is the caller graph for this function:



5.17.3.2 SCREEN_STRUCT* gui_screen_get_current()

Returns the currently active screen

Returns

A Pointer to the active SCREEN_STRUCT

Definition at line 35 of file [screen.c](#).

```

00036 {
00037     return screen_current;
00038 }

```

5.17.3.3 bool gui_screen_navigate (SCREEN_STRUCT * screen)

Navigate to the given screen as soon as the app enters the main loop again (and [gui_screen_update\(\)](#) is called) It's safe to call this method from an interrupt

Note

Do not pass a screen which is already in your history of screens!

Parameters

<i>screen</i>	A Pointer to the preinitialized SCREEN_STRUCT
---------------	---

Returns

true on success

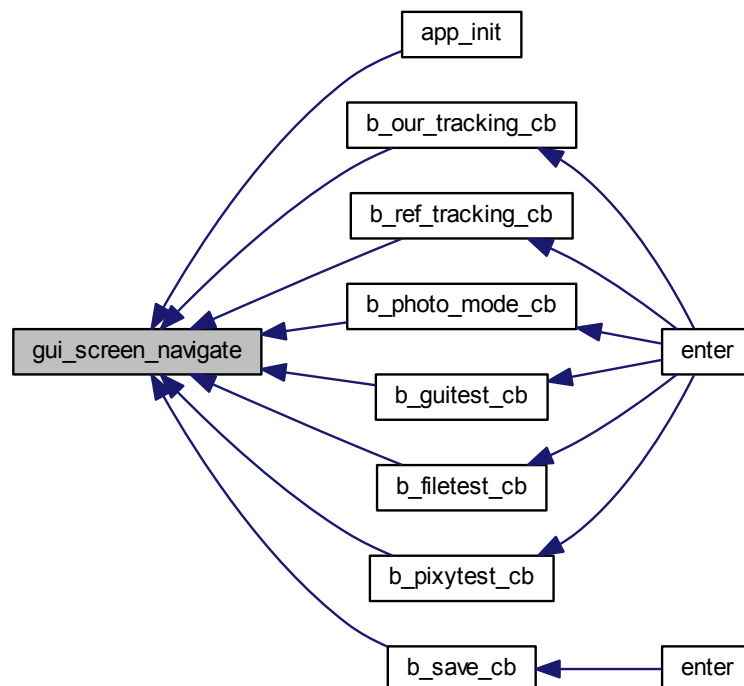
Definition at line 74 of file [screen.c](#).

```

00075 {
00076     if (screen == NULL || screen == screen_current || screen ==
screen_goto) { //invalid argument passed
00077         return false;
00078     }
00079
00080     screen->next = NULL; //this will become the new tail of the list, so the next pointer must be NULL
00081     screen_goto = screen; // "send message" to main loop, to switch the screen
00082     return true;
00083 }

```

Here is the caller graph for this function:



5.17.3.4 void gui_screen_update ()

Updates the current screen. Switches the screen if [gui_screen_navigate\(\)](#) or [gui_screen_back\(\)](#) have been called since the last call to this method. This method should be called repeatedly from the main loop (e.g. [app_process\(\)](#))

Definition at line 41 of file [screen.c](#).

```

00042 {
00043     if (screen_goto != NULL) { //we received the task to switch the screen
00044         SCREEN_STRUCT* go = (SCREEN_STRUCT*)
screen_goto; //Backup volatile variable
00045         screen_goto = NULL; //reset the "goto instruction", since we're processing it now
00046
00047         if (go->next != NULL) { //The screen is not the last in the list, so we're going back
00048             if (go->next != screen_current) { //this condition should always be false
00049                 return; //list corrupted?
00050             }
00051
00052             screen_current->on_leave(screen_current); //let the current
screen free/unregister it's resources
00053             go->next = NULL; //remove the current screen from the list
00054         } else { //we're going forward (to a new screen)
00055             if (screen_current != NULL) { //this is not the first screen
00056                 screen_current->on_leave(screen_current); //let the
current screen free/unregister it's resources
00057                 screen_current->next = go; //append the new screen to the end of the list
00058             } else { //first screen ever seen
00059                 screen_list = go; //set the new screen as list-head
00060             }
00061         }
00062
00063         go->on_enter(go); //let the new screen allocate/register it's resources
00064         screen_current = go; //the new screen is now the current screen. Transition done
00065     }
00066
00067     if (screen_current != NULL) { //A screen has been set
00068         screen_current->on_update(screen_current); //Update current
screen
00069     }
00070 }

```

Here is the caller graph for this function:



5.18 Filesystem (LowLevel)

Collaboration diagram for Filesystem (LowLevel):



Functions

- `bool ll_filesystem_init ()`
- `DIRECTORY_STRUCT * ll_filesystem_dir_open (const char *path)`
- `void ll_filesystem_dir_close (DIRECTORY_STRUCT *dir)`
- `FILE_HANDLE * ll_filesystem_file_open (const char *filename)`
- `void ll_filesystem_file_close (FILE_HANDLE *handle)`
- `FILE_STATUS ll_filesystem_file_seek (FILE_HANDLE *handle, uint32_t offset)`
- `FILE_STATUS ll_filesystem_file_read (FILE_HANDLE *handle, uint8_t *buf, uint32_t size)`
- `FILE_STATUS ll_filesystem_file_write (FILE_HANDLE *handle, uint8_t *buf, uint32_t size)`

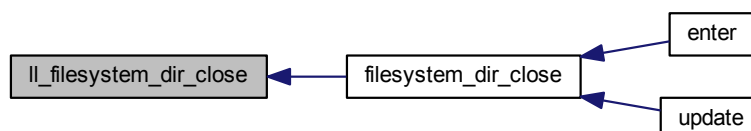
5.18.1 Detailed Description

Low level functions for the `Filesystem` module

5.18.2 Function Documentation

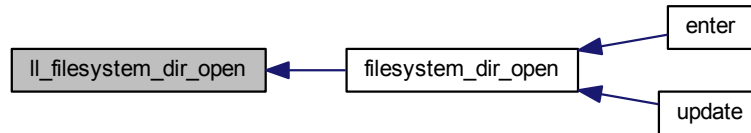
5.18.2.1 `void ll_filesystem_dir_close ( DIRECTORY_STRUCT * dir )`

Here is the caller graph for this function:



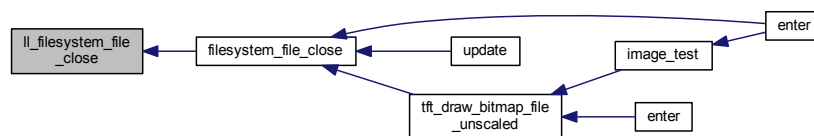
5.18.2.2 DIRECTORY_STRUCT* ll_filesystem_dir_open (const char * path)

Here is the caller graph for this function:



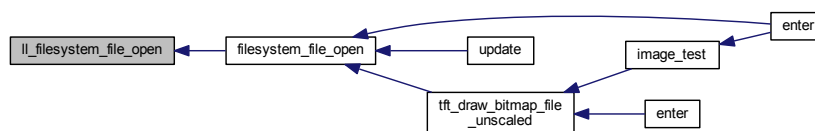
5.18.2.3 void ll_filesystem_file_close (FILE_HANDLE * handle)

Here is the caller graph for this function:



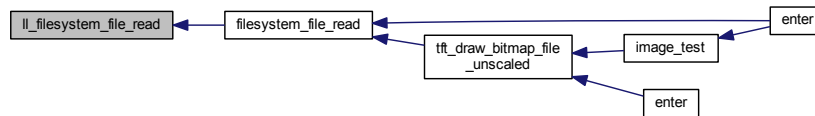
5.18.2.4 FILE_HANDLE* ll_filesystem_file_open (const char * filename)

Here is the caller graph for this function:



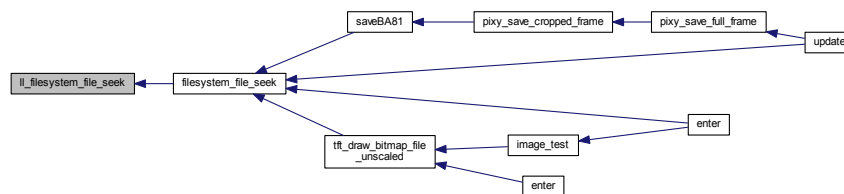
5.18.2.5 FILE_STATUS ll_filesystem_file_read (FILE_HANDLE * handle, uint8_t * buf, uint32_t size)

Here is the caller graph for this function:



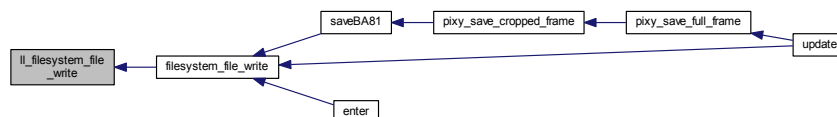
5.18.2.6 FILE_STATUS ll_filesystem_file_seek (FILE_HANDLE * handle, uint32_t offset)

Here is the caller graph for this function:



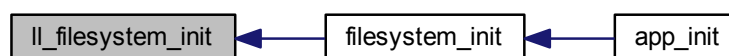
5.18.2.7 FILE_STATUS ll_filesystem_file_write (FILE_HANDLE * handle, uint8_t * buf, uint32_t size)

Here is the caller graph for this function:



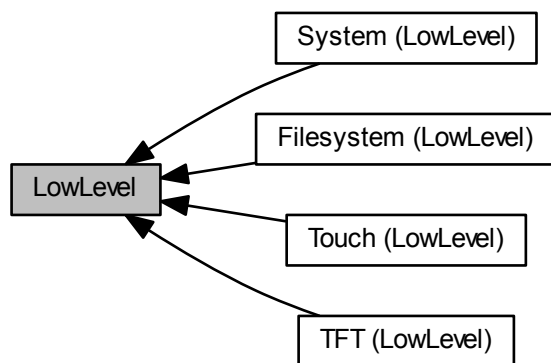
5.18.2.8 bool ll_filesystem_init ()

Here is the caller graph for this function:



5.19 LowLevel

Collaboration diagram for LowLevel:



Modules

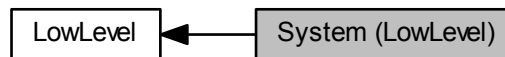
- [Filesystem \(LowLevel\)](#)
- [System \(LowLevel\)](#)
- [TFT \(LowLevel\)](#)
- [Touch \(LowLevel\)](#)

5.19.1 Detailed Description

The Low-Level platform abstraction layer

5.20 System (LowLevel)

Collaboration diagram for System (LowLevel):



Functions

- bool [ll_system_init](#) ()
- void [ll_system_delay](#) (uint32_t msec)
- void [ll_system_process](#) ()
- void [ll_system_toggle_led](#) ()

5.20.1 Detailed Description

Low level functions of the [System](#) Module

5.20.2 Function Documentation

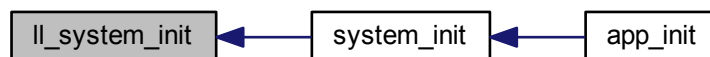
5.20.2.1 void [ll_system_delay](#) (uint32_t msec)

Here is the caller graph for this function:



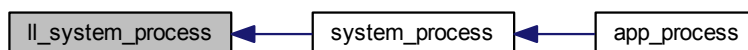
5.20.2.2 `bool ll_system_init ( )`

Here is the caller graph for this function:



5.20.2.3 `void ll_system_process ( )`

Here is the caller graph for this function:



5.20.2.4 `void ll_system_toggle_led ( )`

Here is the caller graph for this function:



5.21 TFT (LowLevel)

Collaboration diagram for TFT (LowLevel):



Functions

- bool [ll_tft_init](#) ()
- void [ll_tft_clear](#) (uint16_t color)
- void [ll_tft_draw_line](#) (uint16_t x1, uint16_t y1, uint16_t x2, uint16_t y2, uint16_t color)
- void [ll_tft_draw_pixel](#) (uint16_t x, uint16_t y, uint16_t color)
- void [ll_tft_draw_rectangle](#) (uint16_t x1, uint16_t y1, uint16_t x2, uint16_t y2, uint16_t color)
- void [ll_tft_fill_rectangle](#) (uint16_t x1, uint16_t y1, uint16_t x2, uint16_t y2, uint16_t color)
- void [ll_tft_draw_bitmap_unscaled](#) (uint16_t x, uint16_t y, uint16_t width, uint16_t height, const uint16_t *dat)
- void [ll_tft_draw_circle](#) (uint16_t x, uint16_t y, uint16_t r, uint16_t color)
- uint8_t [ll_tft_num_fonts](#) ()
- uint8_t [ll_tft_font_height](#) (uint8_t fontnum)
- uint8_t [ll_tft_font_width](#) (uint8_t fontnum)
- void [ll_tft_draw_char](#) (uint16_t x, uint16_t y, uint16_t color, uint16_t bgcolor, uint8_t font, char c)

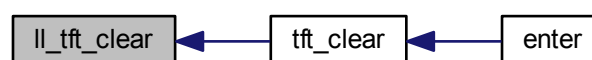
5.21.1 Detailed Description

Low level functions for the [TFT](#) module

5.21.2 Function Documentation

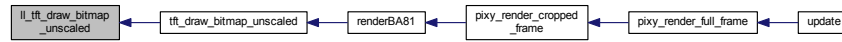
5.21.2.1 void [ll_tft_clear](#) (uint16_t color)

Here is the caller graph for this function:



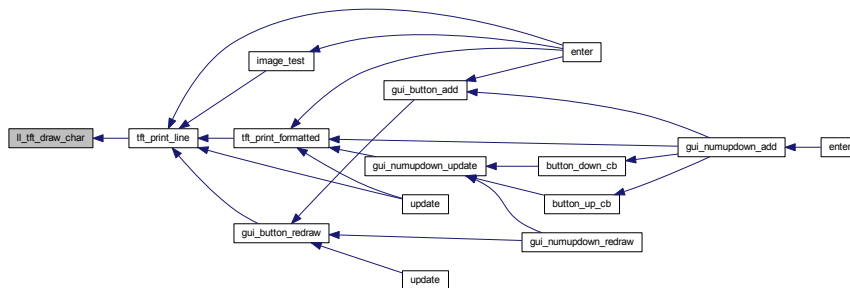
5.21.2.2 void ll_tft_draw_bitmap_unscaled (uint16_t x, uint16_t y, uint16_t width, uint16_t height, const uint16_t * dat)

Here is the caller graph for this function:



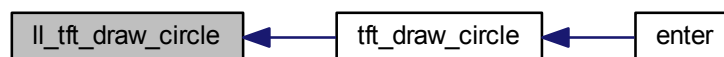
5.21.2.3 void ll_tft_draw_char (uint16_t x, uint16_t y, uint16_t color, uint16_t bgcolor, uint8_t font, char c)

Here is the caller graph for this function:



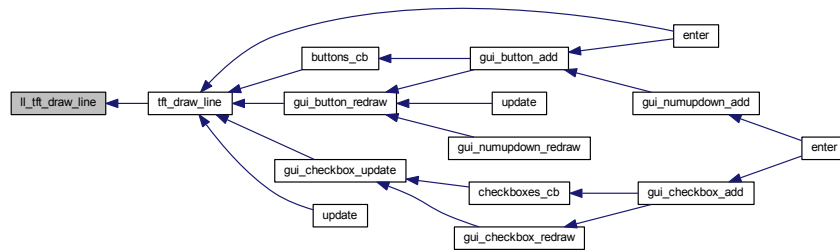
5.21.2.4 void ll_tft_draw_circle (uint16_t x, uint16_t y, uint16_t r, uint16_t color)

Here is the caller graph for this function:



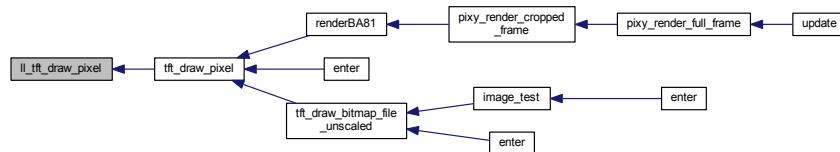
5.21.2.5 void ll_tft_draw_line (uint16_t x1, uint16_t y1, uint16_t x2, uint16_t y2, uint16_t color)

Here is the caller graph for this function:



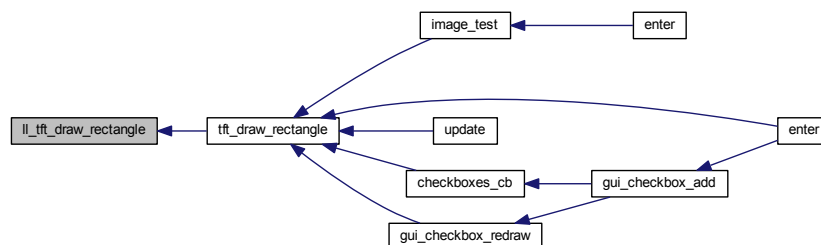
5.21.2.6 void ll_tft_draw_pixel (uint16_t x, uint16_t y, uint16_t color)

Here is the caller graph for this function:



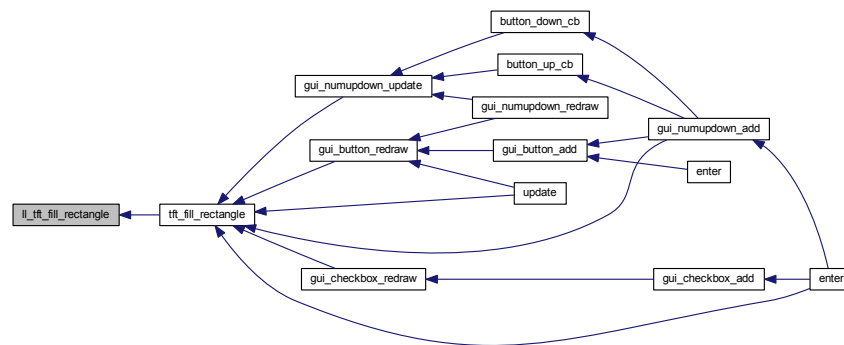
5.21.2.7 void ll_tft_draw_rectangle (uint16_t x1, uint16_t y1, uint16_t x2, uint16_t y2, uint16_t color)

Here is the caller graph for this function:



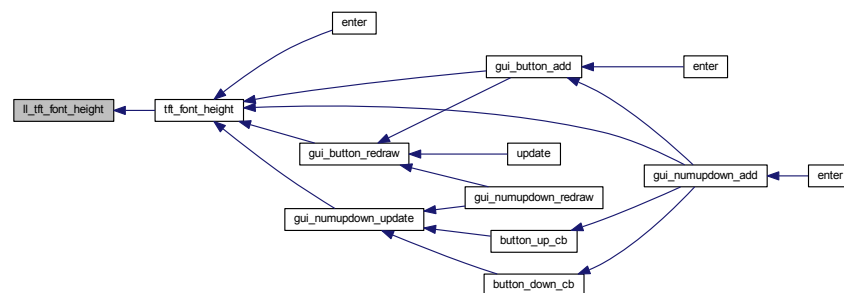
5.21.2.8 void ll_tft_fill_rectangle (uint16_t x1, uint16_t y1, uint16_t x2, uint16_t y2, uint16_t color)

Here is the caller graph for this function:



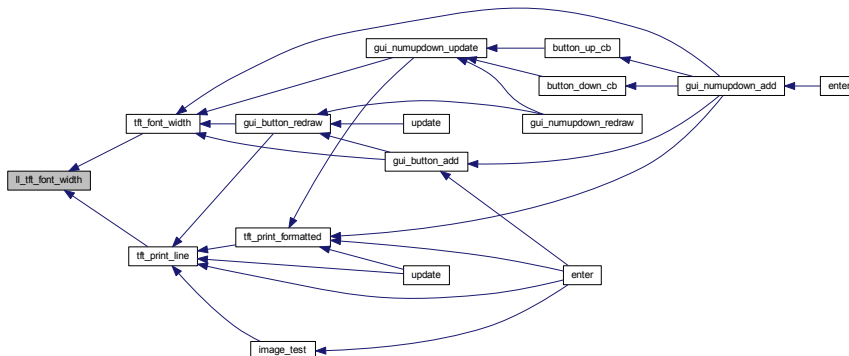
5.21.2.9 uint8_t ll_tft_font_height (uint8_t fontnum)

Here is the caller graph for this function:



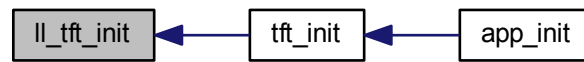
5.21.2.10 uint8_t ll_tft_font_width (uint8_t fontnum)

Here is the caller graph for this function:

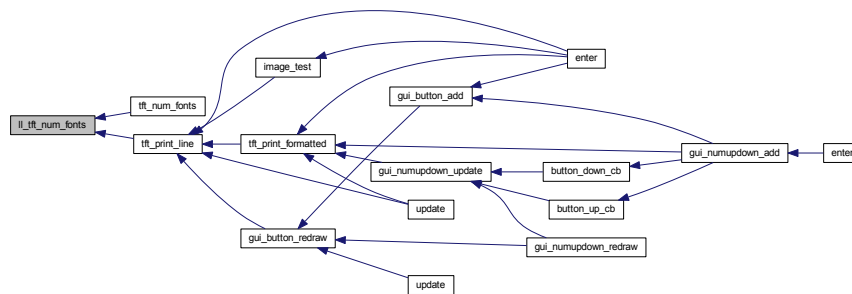


5.21.2.11 `bool ll_tft_init ( )`

Here is the caller graph for this function:

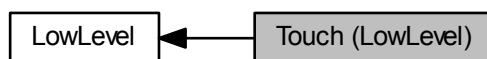
5.21.2.12 `uint8_t ll_tft_num_fonts ( )`

Here is the caller graph for this function:



5.22 Touch (LowLevel)

Collaboration diagram for Touch (LowLevel):



Functions

- bool [ll_touch_init](#) ()

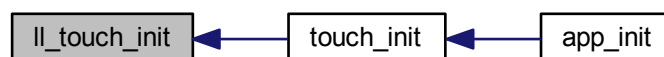
5.22.1 Detailed Description

Low level functions for the [Touch](#) module

5.22.2 Function Documentation

5.22.2.1 bool ll_touch_init ()

Here is the caller graph for this function:



5.23 Pixy

Data Structures

- struct [Block](#)

Macros

- `#define PIXY_MAX_SIGNATURE 7`
- `#define PIXY_MIN_X 0`
- `#define PIXY_MAX_X 319`
- `#define PIXY_MIN_Y 0`
- `#define PIXY_MAX_Y 199`
- `#define PIXY_RCS_MIN_POS 0`
- `#define PIXY_RCS_MAX_POS 1000`
- `#define PIXY_RCS_CENTER_POS ((PIXY_RCS_MAX_POS-PIXY_RCS_MIN_POS)/2)`
- `#define PIXY_BLOCKTYPE_NORMAL 0`
- `#define PIXY_BLOCKTYPE_COLOR_CODE 1`

Functions

- `int pixy_init ()`
Creates a connection with Pixy and listens for Pixy messages.
- `int pixy_blocks_are_new ()`
Indicates when new block data from Pixy is received.
- `int pixy_get_blocks (uint16_t max_blocks, struct Block *blocks)`
Copies up to 'max_blocks' number of Blocks to the address pointed to by 'blocks'.
- `int pixy_service ()`
- `int pixy_command (const char *name,...)`
Send a command to Pixy.
- `void pixy_close ()`
Terminates connection with Pixy.
- `void pixy_error (int error_code)`
Send description of pixy error to stdout.
- `int pixy_led_set_RGB (uint8_t red, uint8_t green, uint8_t blue)`
Set color of pixy LED.
- `int pixy_led_set_max_current (uint32_t current)`
Set pixy LED maximum current.
- `int pixy_led_get_max_current ()`
Get pixy LED maximum current.
- `int pixy_cam_set_auto_white_balance (uint8_t value)`
Enable or disable pixy camera auto white balance.
- `int pixy_cam_get_auto_white_balance ()`
Get pixy camera auto white balance setting.
- `uint32_t pixy_cam_get_white_balance_value ()`
Get pixy camera white balance()
- `int pixy_cam_set_white_balance_value (uint8_t red, uint8_t green, uint8_t blue)`
Set pixy camera white balance.
- `int pixy_cam_set_auto_exposure_compensation (uint8_t enable)`
Enable or disable pixy camera auto exposure compensation.
- `int pixy_cam_get_auto_exposure_compensation ()`

- Get pixy camera auto exposure compensation setting.*

 - int [pixy_cam_set_exposure_compensation](#) (uint8_t gain, uint16_t comp)

Set pixy camera exposure compensation.
- int [pixy_cam_get_exposure_compensation](#) (uint8_t *gain, uint16_t *comp)

Get pixy camera exposure compensation.
- int [pixy_cam_set_brightness](#) (uint8_t brightness)

Set pixy camera brightness.
- int [pixy_cam_get_brightness](#) ()

Get pixy camera brightness.
- int [pixy_rcs_get_position](#) (uint8_t channel)

Get pixy servo axis position.
- int [pixy_rcs_set_position](#) (uint8_t channel, uint16_t position)

Set pixy servo axis position.
- int [pixy_rcs_set_frequency](#) (uint16_t frequency)

Set pixy servo pulse width modulation (PWM) frequency.
- int [pixy_get_firmware_version](#) (uint16_t *major, uint16_t *minor, uint16_t *build)

Get pixy firmware version.

5.23.1 Detailed Description

The Pixy Module

5.23.2 Macro Definition Documentation

5.23.2.1 #define PIXY_BLOCKTYPE_COLOR_CODE 1

Definition at line 51 of file [pixy.h](#).

5.23.2.2 #define PIXY_BLOCKTYPE_NORMAL 0

Definition at line 50 of file [pixy.h](#).

5.23.2.3 #define PIXY_MAX_SIGNATURE 7

Definition at line 36 of file [pixy.h](#).

5.23.2.4 #define PIXY_MAX_X 319

Definition at line 40 of file [pixy.h](#).

5.23.2.5 #define PIXY_MAX_Y 199

Definition at line 42 of file [pixy.h](#).

5.23.2.6 #define PIXY_MIN_X 0

Definition at line 39 of file [pixy.h](#).

5.23.2.7 `#define PIXY_MIN_Y 0`

Definition at line 41 of file [pixy.h](#).

5.23.2.8 `#define PIXY_RCS_CENTER_POS ((PIXY_RCS_MAX_POS-PIXY_RCS_MIN_POS)/2)`

Definition at line 47 of file [pixy.h](#).

5.23.2.9 `#define PIXY_RCS_MAX_POS 1000`

Definition at line 46 of file [pixy.h](#).

5.23.2.10 `#define PIXY_RCS_MIN_POS 0`

Definition at line 45 of file [pixy.h](#).

5.23.3 Function Documentation

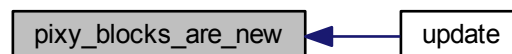
5.23.3.1 `int pixy_blocks_are_new ( )`

Indicates when new block data from Pixy is received.

Returns

- 1 New Data: [Block](#) data has been updated.
- 0 Stale Data: [Block](#) data has not changed since [pixy_get_blocks\(\)](#) was last called.

Here is the caller graph for this function:



5.23.3.2 `int pixy_cam_get_auto_exposure_compensation ( )`

Get pixy camera auto exposure compensation setting.

Returns

- 1 Auto exposure compensation enabled.
- 0 Auto exposure compensation disabled.
- Negative Error

5.23.3.3 `int pixy_cam_get_auto_white_balance ( )`

Get pixy camera auto white balance setting.

Returns

1 Auto white balance is enabled.
0 Auto white balance is disabled.
Negative Error

5.23.3.4 `int pixy_cam_get_brightness ( )`

Get pixy camera brightness.

Returns

Non-negative Brightness value.
Negative Error

5.23.3.5 `int pixy_cam_get_exposure_compensation ( uint8_t * gain, uint16_t * comp )`

Get pixy camera exposure compensation.

Parameters

out	<i>gain</i>	Camera gain.
out	<i>comp</i>	Camera exposure compensation.

Returns

0 Success
Negative Error

5.23.3.6 `uint32_t pixy_cam_get_white_balance_value ( )`

Get pixy camera white balance()

Returns

Composite value for RGB white balance: white balance = green_value + (red_value << 8) + (blue << 16)

5.23.3.7 `int pixy_cam_set_auto_exposure_compensation ( uint8_t enable )`

Enable or disable pixy camera auto exposure compensation.

Parameters

in	<i>enable</i>	0: Disable auto exposure compensation. 1: Enable auto exposure compensation.
----	---------------	--

Returns

0 Success
Negative Error

5.23.3.8 `int pixy_cam_set_auto_white_balance ( uint8_t value )`

Enable or disable pixy camera auto white balance.

Parameters

<i>value</i>	1: Enable white balance. 0: Disable white balance.
--------------	--

Returns

0 Success
Negative Error

5.23.3.9 int pixy_cam_set_brightness (uint8_t *brightness*)

Set pixy camera brightness.

Parameters

<i>in</i>	<i>brightness</i>	Brightness value.
-----------	-------------------	-------------------

Returns

0 Success
Negative Error

5.23.3.10 int pixy_cam_set_exposure_compensation (uint8_t *gain*, uint16_t *comp*)

Set pixy camera exposure compensation.

Parameters

<i>in</i>	<i>gain</i>	Camera gain.
<i>in</i>	<i>comp</i>	Camera exposure compensation.

Returns

0 Success
Negative Error

5.23.3.11 int pixy_cam_set_white_balance_value (uint8_t *red*, uint8_t *green*, uint8_t *blue*)

Set pixy camera white balance.

Parameters

<i>in</i>	<i>red</i>	Red white balance value.
<i>in</i>	<i>green</i>	Green white balance value.
<i>in</i>	<i>blue</i>	Blue white balance value.

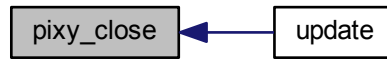
Returns

0 Success
Negative Error

5.23.3.12 void pixy_close ()

Terminates connection with Pixy.

Here is the caller graph for this function:



5.23.3.13 `int pixy_command ( const char * name, ... )`

Send a command to Pixy.

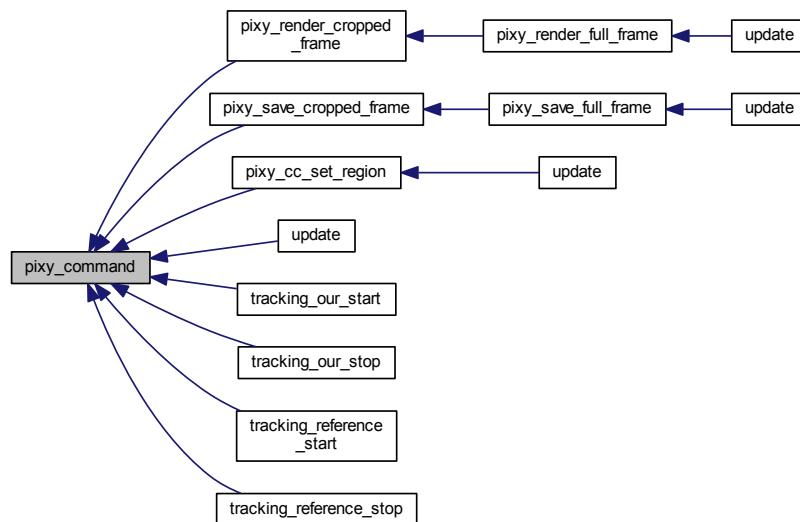
Parameters

<code>in</code>	<code>name</code>	Chirp remote procedure call identifier string.
-----------------	-------------------	--

Returns

-1 Error

Here is the caller graph for this function:



5.23.3.14 `void pixy_error ( int error_code )`

Send description of pixy error to stdout.

Parameters

in	<i>error_code</i>	Pixy error code
----	-------------------	-----------------

5.23.3.15 int pixy_get_blocks (uint16_t *max_blocks*, struct Block * *blocks*)

Copies up to 'max_blocks' number of Blocks to the address pointed to by 'blocks'.

Parameters

in	<i>max_blocks</i>	Maximum number of Blocks to copy to the address pointed to by 'blocks'.
out	<i>blocks</i>	Address of an array in which to copy the blocks to. The array must be large enough to write 'max_blocks' number of Blocks to.

Returns

Non-negative Success: Number of blocks copied
 PIXY_ERROR_USB_IO USB Error: I/O
 PIXY_ERROR_NOT_FOUND USB Error: Pixy not found
 PIXY_ERROR_USB_BUSY USB Error: Busy
 PIXY_ERROR_USB_NO_DEVICE USB Error: No device
 PIXY_ERROR_INVALID_PARAMETER Invalid parameter specified

Here is the caller graph for this function:

5.23.3.16 int pixy_get_firmware_version (uint16_t * *major*, uint16_t * *minor*, uint16_t * *build*)

Get pixy firmware version.

Parameters

out	<i>major</i>	Major version component
out	<i>minor</i>	Minor version component
out	<i>build</i>	Build identifier

Returns

0 Success
 Negative Error

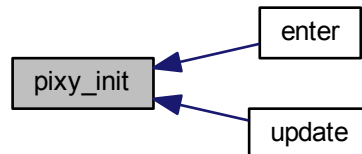
5.23.3.17 int pixy_init ()

Creates a connection with Pixy and listens for Pixy messages.

Returns

0 Success
 PIXY_ERROR_USB_IO USB Error: I/O
 PIXY_ERROR_NOT_FOUND USB Error: Pixy not found
 PIXY_ERROR_USB_BUSY USB Error: Busy
 PIXY_ERROR_USB_NO_DEVICE USB Error: No device

Here is the caller graph for this function:

5.23.3.18 `int pixy_led_get_max_current ( )`

Get pixy LED maximum current.

Returns

Non-negative Maximum LED current value (microamps).
 Negative Error

5.23.3.19 `int pixy_led_set_max_current ( uint32_t current )`

Set pixy LED maximum current.

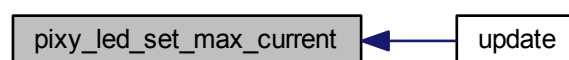
Parameters

<code>in</code>	<code>current</code>	Maximum current (microamps).
-----------------	----------------------	------------------------------

Returns

0 Success
 Negative Error

Here is the caller graph for this function:



5.23.3.20 `int pixy_led_set_RGB ( uint8_t red, uint8_t green, uint8_t blue )`

Set color of pixy LED.

Parameters

in	<i>red</i>	Brightness value for red LED element. [0, 255] 0 = Off, 255 = On
in	<i>green</i>	Brightness value for green LED element. [0, 255] 0 = Off, 255 = On
in	<i>blue</i>	Brightness value for blue LED element. [0, 255] 0 = Off, 255 = On

Returns

0 Success
Negative Error

Here is the caller graph for this function:

5.23.3.21 int pixy_rcs_get_position (uint8_t *channel*)

Get pixy servo axis position.

Parameters

<i>channel</i>	Channel value. Range: [0, 1]
----------------	------------------------------

Returns

Position of channel. Range: [0, 999]
Negative Error

5.23.3.22 int pixy_rcs_set_frequency (uint16_t *frequency*)

Set pixy servo pulse width modulation (PWM) frequency.

Parameters

<i>frequency</i>	Range: [20, 300] Hz Default: 50 Hz
------------------	------------------------------------

5.23.3.23 int pixy_rcs_set_position (uint8_t *channel*, uint16_t *position*)

Set pixy servo axis position.

Parameters

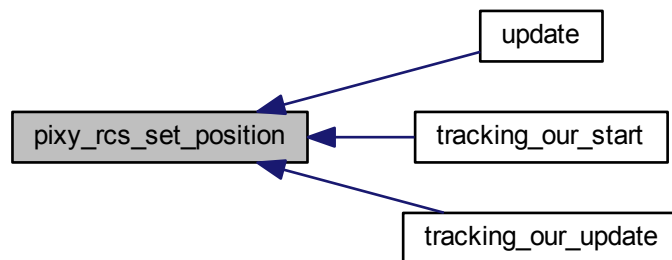
<i>channel</i>	Channel value. Range: [0, 1]
----------------	------------------------------

<i>position</i>	Position value of the channel. Range: [0, 999]
-----------------	--

Returns

0 Success
Negative Error

Here is the caller graph for this function:

**5.23.3.24 int pixy_service ()**

Here is the caller graph for this function:



5.24 System

Functions

- bool [system_init](#) ()
- void [system_delay](#) (uint32_t msec)
- void [system_process](#) ()
- void [system_toggle_led](#) ()

5.24.1 Detailed Description

The System Module provides access to delay functions, leds and provides a system init function

5.24.2 Function Documentation

5.24.2.1 void system_delay (uint32_t msec)

Sleeps for a certain amount of time

Parameters

<i>msec</i>	The number of milliseconds to sleep
-------------	-------------------------------------

Definition at line 26 of file [system.c](#).

```
00027 {
00028     ll_system_delay(msec);
00029 }
```

Here is the call graph for this function:



5.24.2.2 bool system_init ()

Initializes the system. Call this method at the start of your [app_init\(\)](#) function and before using any system_* functions

Returns

true on success

Definition at line 21 of file [system.c](#).

```
00022 {
00023     return ll_system_init();
00024 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.24.2.3 void system_process ()

Executes pending system events (like handling usb, timers etc). Call this somewhere in [app_process\(\)](#).

Definition at line 31 of file [system.c](#).

```
00032 {  
00033     ll_system_process();  
00034 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.24.2.4 void system_toggle_led ()

Toggles a Status Led. Use this function for debugging or to show activity

Definition at line 36 of file [system.c](#).

```
00037 {  
00038     ll_system_toggle_led();  
00039 }
```

Here is the call graph for this function:



5.25 TFT

Macros

- `#define RGB(r, g, b) (((r) & 0xF8) << 8) | (((g) & 0xFC) << 3) | (((b) & 0xF8) >> 3)`
- `#define RED RGB(255,0,0)`
- `#define GREEN RGB(0,255,0)`
- `#define BLUE RGB(0,0,255)`
- `#define WHITE 0xF7BE`
- `#define BLACK RGB(0,0,0)`
- `#define HEX(h) (RGB(((h)>>16),((h)>>8),(h)))`
- `#define TRANSPARENT ((uint16_t)0x80C2)`

Functions

- `bool tft_init ()`
- `void tft_clear (uint16_t color)`
- `void tft_draw_line (uint16_t x1, uint16_t y1, uint16_t x2, uint16_t y2, uint16_t color)`
- `void tft_draw_pixel (uint16_t x, uint16_t y, uint16_t color)`
- `void tft_draw_rectangle (uint16_t x1, uint16_t y1, uint16_t x2, uint16_t y2, uint16_t color)`
- `void tft_fill_rectangle (uint16_t x1, uint16_t y1, uint16_t x2, uint16_t y2, uint16_t color)`
- `void tft_draw_bitmap_unscaled (uint16_t x, uint16_t y, uint16_t width, uint16_t height, const uint16_t *dat)`
- `bool tft_draw_bitmap_file_unscaled (uint16_t x, uint16_t y, const char *filename)`
- `void tft_draw_circle (uint16_t x, uint16_t y, uint16_t r, uint16_t color)`
- `uint8_t tft_num_fonts ()`
- `uint8_t tft_font_height (uint8_t fontnum)`
- `uint8_t tft_font_width (uint8_t fontnum)`
- `void tft_print_line (uint16_t x, uint16_t y, uint16_t color, uint16_t bgcolor, uint8_t font, const char *text)`
- `void tft_print_formatted (uint16_t x, uint16_t y, uint16_t color, uint16_t bgcolor, uint8_t font, const char *format,...)`

5.25.1 Detailed Description

The TFT Modul provides access to the display

5.25.2 Macro Definition Documentation

5.25.2.1 `#define BLACK RGB(0,0,0)`

Definition at line 54 of file [tft.h](#).

5.25.2.2 `#define BLUE RGB(0,0,255)`

Definition at line 52 of file [tft.h](#).

5.25.2.3 `#define GREEN RGB(0,255,0)`

Definition at line 51 of file [tft.h](#).

5.25.2.4 `#define HEX( h )(RGB(((h)>>16),((h)>>8),(h)))`

Creates a 16bit color from a 24bit hex rgb color code

Returns

Definition at line 60 of file [tft.h](#).

5.25.2.5 `#define RED RGB(255,0,0)`

Definition at line 50 of file [tft.h](#).

5.25.2.6 `#define RGB( r, g, b ) (((r) & 0xF8) << 8) | (((g) & 0xFC) << 3) | (((b) & 0xF8) >> 3)`

Creates a 16bit color from 8bit * 3 colors (r,g,b)

Returns

Definition at line 48 of file [tft.h](#).

5.25.2.7 `#define TRANSPARENT ((uint16_t)0x80C2)`

Transparent color

Returns

Definition at line 66 of file [tft.h](#).

5.25.2.8 `#define WHITE 0xF7BE`

Definition at line 53 of file [tft.h](#).

5.25.3 Function Documentation

5.25.3.1 `void tft_clear ( uint16_t color )`

Clears the entire display with the given color. Overpaints everything which was there before.

Parameters

<i>color</i>	The 16-bit color to clear the display with.
--------------	---

Definition at line 45 of file [tft.c](#).

```
00046 {
00047     ll_tft_clear(color);
00048 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.25.3.2 bool tft_draw_bitmap_file_unscaled (uint16_t x, uint16_t y, const char * filename)

Draws a bitmap from the filesystem onto the display without scaling/cropping. The bitmap must be saved in the windows bitmap format (.bmp) without compression and with 24 (b,g,r) or 32 (a,r,g,b) bits per pixel.

Parameters

<i>x</i>	The x-coordinate of the top-left corner to draw the bitmap at
<i>y</i>	The y-coordinate of the top-left corner to draw the bitmap at
<i>filename</i>	The absolute path to the .bmp file

Returns

true on success

Definition at line 123 of file [tft.c](#).

```

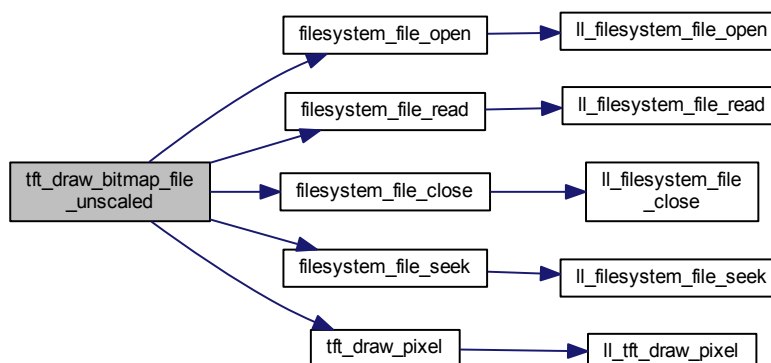
00124 {
00125     //This method reads a .bmp file from the filesystem and tries to draw it.
00126     //Note: The bmp implementation is not complete, it has some limitations and it makes assumptions. See
doxygen comment for this method.
00127     //Source Copied and adapted from: http://stackoverflow.com/a/17040962/2606757
00128
00129     FILE_HANDLE* file = filesystem_file_open(filename); //try to open the
file
00130
00131     if (file == NULL) { //file opening failed
00132         return false;
00133     }
00134
00135     unsigned char info[54];
00136
00137     if (filesystem_file_read(file, info, 54) != F_OK) { //try to read the 54 byte
header
00138         filesystem_file_close(file);
00139         return false; //reading the header failed
00140     }
00141
00142     // extract image height and width from header
  
```

```

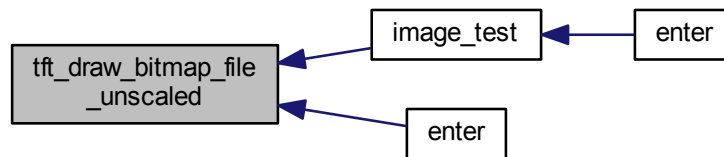
00143     uint32_t width = *(uint32_t*)&info[18]; //width in pixel
00144     uint32_t height = *(uint32_t*)&info[22]; //height in pixel
00145     uint16_t depth = *(uint16_t*)&info[28]; //bit's per pixel (color depth)
00146     depth /= 8; //we want the number of bytes per pixel
00147
00148     filesystem_file_seek(file, *(uint32_t*)&info[10]); //seek to the place where img
data begins
00149
00150     uint32_t row_padded = (width * depth + 3) & (~3); //row size must be aligned to 4 bytes
00151
00152     unsigned char data [row_padded]; //allocate space for one row (incl. padding)
00153
00154     for (int i = 0; i < height; i++) { //for each row
00155         filesystem_file_read(file, data, row_padded); //read row into buffer
00156
00157         for (int j = 0; j < width * depth; j += depth) { //for each pixel
00158             unsigned char a, r, g, b;
00159
00160             if (depth == 4) { //a,r,g,b 8bit each
00161                 a = data[j];
00162                 r = data[j + 1];
00163                 g = data[j + 2];
00164                 b = data[j + 3];
00165             } else if (depth == 3) { // b,g,r, 8bit each
00166                 a = 255;
00167                 r = data[j + 2];
00168                 g = data[j + 1];
00169                 b = data[j];
00170             }
00171
00172             if (a != 0) {
00173                 //bmp's are stored "bottom-up", so we start drawing at the bottom
00174                 tft_draw_pixel(x + j / depth, y + height - 1 - i,
RGB(r, g, b));
00175             }
00176         }
00177     }
00178
00179     filesystem_file_close(file);
00180
00181     return true;
00182
00183 }

```

Here is the call graph for this function:



Here is the caller graph for this function:



5.25.3.3 `void tft_draw_bitmap_unscaled ( uint16_t x, uint16_t y, uint16_t width, uint16_t height, const uint16_t * dat )`

Draws a bitmap onto the display without scaling/cropping. The bitmap must be provided as an array of 16-bit colors

Parameters

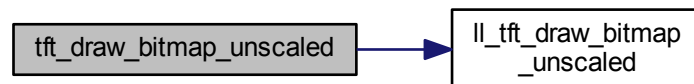
<i>x</i>	The x-coordinate of the top-left corner to draw the bitmap at
<i>y</i>	The y-coordinate of the top-left corner to draw the bitmap at
<i>width</i>	The width of the bitmap in pixels
<i>height</i>	The height of the bitmap in pixels
<i>dat</i>	A pointer to a <code>uint16_t</code> array containing the colors for each pixel. Starting in the topleft and going from left to right, line by line.

Definition at line 72 of file [tft.c](#).

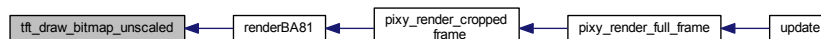
```

00073 {
00074     ll_tft_draw_bitmap_unscaled(x, y, width,
00075     height, dat);
00075 }
  
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.25.3.4 void tft_draw_circle(uint16_t x, uint16_t y, uint16_t r, uint16_t color)

Draws the outline of a circle onto the display

Parameters

<i>x</i>	The x-Coordinate of the center point
<i>y</i>	The y-Coordinate of the center point
<i>r</i>	The Radius in Pixels
<i>color</i>	The 16-Bit color to draw the circle with

Definition at line 77 of file [tft.c](#).

```
00078 {
00079     ll_tft_draw_circle(x, y, r, color);
00080 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.25.3.5 void tft_draw_line (uint16_t x1, uint16_t y1, uint16_t x2, uint16_t y2, uint16_t color)

Draws a line onto the display. The pixels specified by start/end point are inclusive!

Parameters

<i>x1</i>	The x-Coordinate of the start-point
<i>y1</i>	The y-Coordinate of the start-point
<i>x2</i>	The x-Coordinate of the end-point
<i>y2</i>	The y-Coordinate of the end-point
<i>color</i>	The 16-bit color to draw the line with

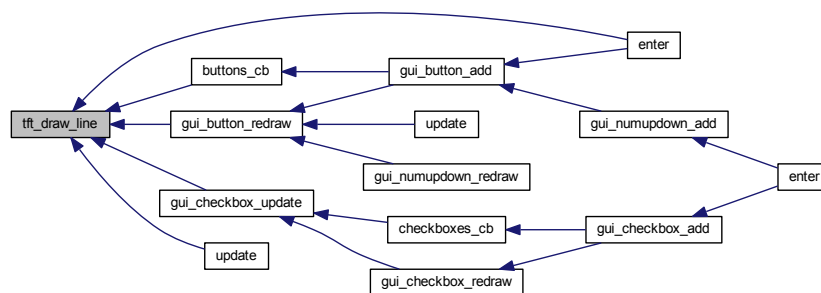
Definition at line 50 of file [tft.c](#).

```
00051 {
00052     ll_tft_draw_line(x1, y1, x2, y2, color);
00053 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.25.3.6 void tft_draw_pixel (uint16_t x, uint16_t y, uint16_t color)

Draws a pixel onto the display.

Parameters

<i>x</i>	The x-Coordinate of the pixel
<i>y</i>	The y-Coordinate of the pixel
<i>color</i>	The 16-bit color to draw the pixel with

Definition at line 56 of file [tft.c](#).

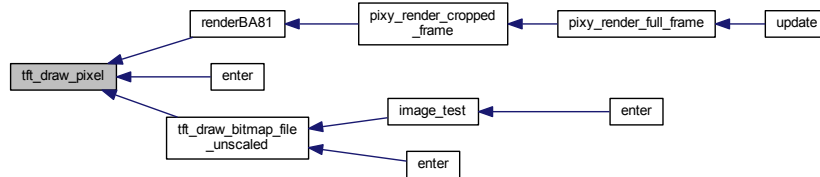
```

00057 {
00058     ll_tft_draw_pixel(x, y, color);
00059 }
  
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.25.3.7 void tft_draw_rectangle (uint16_t x1, uint16_t y1, uint16_t x2, uint16_t y2, uint16_t color)

Draws the outline of a rectangle onto the display. The outline is one pixel wide and goes through the specified start and endpoint.

Parameters

<i>x1</i>	The x-Coordinate of the start-point
<i>y1</i>	The y-Coordinate of the start-point
<i>x2</i>	The x-Coordinate of the end-point
<i>y2</i>	The y-Coordinate of the end-point
<i>color</i>	The 16-bit color to draw the pixel with

Definition at line 61 of file [tft.c](#).

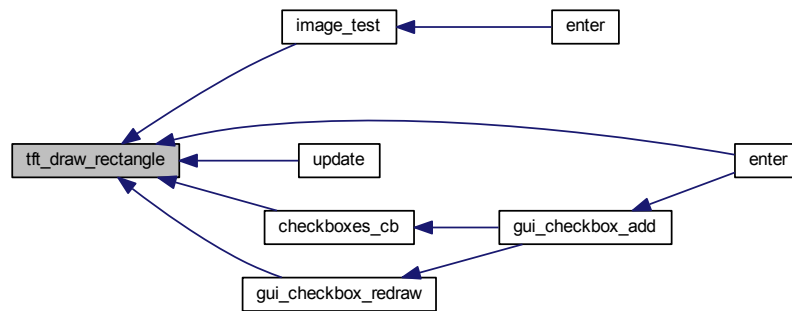
```

00062 {
00063     //could be implemented with 4 lines instead of introducing a ll func
00064     ll_tft_draw_rectangle(x1, y1, x2, y2, color);
00065 }
  
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.25.3.8 void tft_fill_rectangle (uint16_t x1, uint16_t y1, uint16_t x2, uint16_t y2, uint16_t color)

Draws a filled rectangle onto the display. The start,end points are inclusive

Parameters

<i>x1</i>	The x-Coordinate of the start-point
<i>y1</i>	The y-Coordinate of the start-point
<i>x2</i>	The x-Coordinate of the end-point
<i>y2</i>	The y-Coordinate of the end-point
<i>color</i>	The 16-bit color to draw the pixel with

Definition at line 67 of file [tft.c](#).

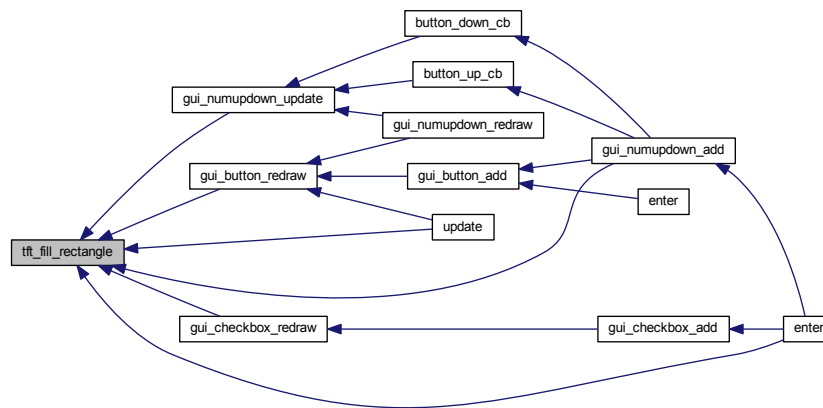
```

00068 {
00069     ll_tft_fill_rectangle(x1, y1, x2, y2, color);
00070 }
  
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.25.3.9 uint8_t tft_font_height (uint8_t fontnum)

Get the height of a font

Parameters

<i>fontnum</i>	The number of the font, from 0 .. (num_fonts -1)
----------------	--

Returns

The height in pixel

Definition at line 87 of file [tft.c](#).

```

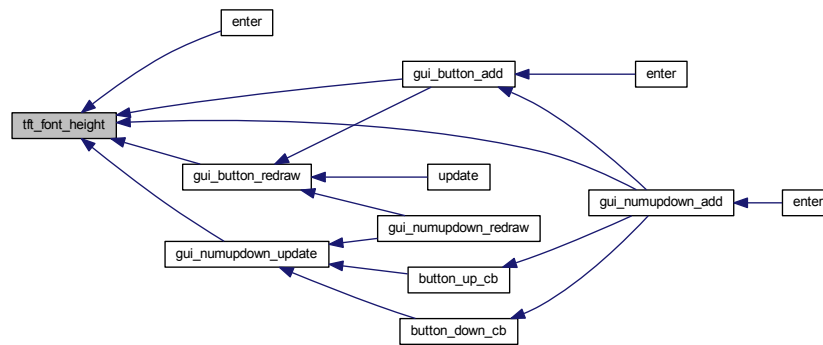
00088 {
00089     return ll_tft_font_height(fontnum);
00090 }

```

Here is the call graph for this function:



Here is the caller graph for this function:



5.25.3.10 uint8_t tft_font_width (uint8_t fontnum)

Get the width of a font

Parameters

<i>fontnum</i>	The number of the font, from 0 .. (num_fonts -1)
----------------	--

Returns

The width in pixel

Definition at line 92 of file [tft.c](#).

```

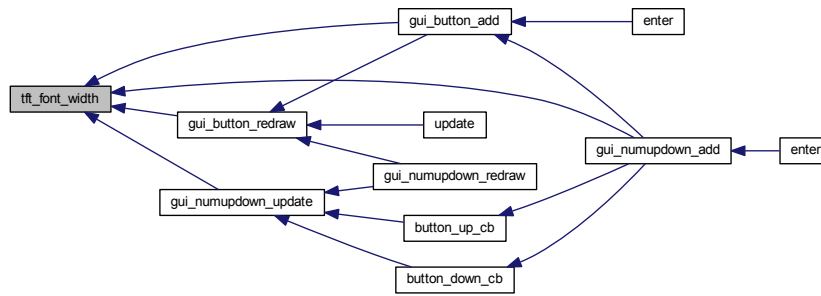
00093 {
00094     return ll_tft_font_width(fontnum);
00095 }

```

Here is the call graph for this function:



Here is the caller graph for this function:



5.25.3.11 bool tft_init ()

Initializes the display. Call this method before using any tft_* functions

Returns

true on success

Definition at line 39 of file [tft.c](#).

```

00040 {
00041     return ll_tft_init();
00042 }
00043 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.25.3.12 `uint8_t tft_num_fonts ( )`

Queries the number of available fonts

Returns

Definition at line 82 of file [tft.c](#).

```
00083 {
00084     return ll_tft_num_fonts();
00085 }
```

Here is the call graph for this function:

5.25.3.13 `void tft_print_formatted ( uint16_t x, uint16_t y, uint16_t color, uint16_t bgcolor, uint8_t font, const char * format, ... )`

Prints a formatted text (like printf) onto the display

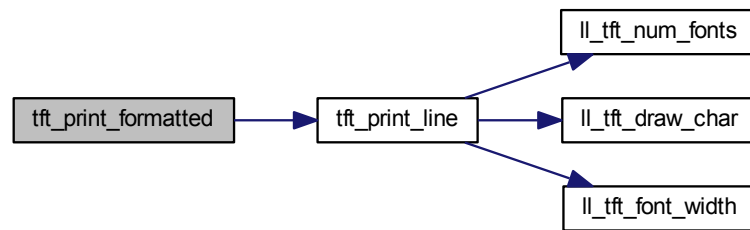
Parameters

<i>x</i>	The x-Coordinate of the Top-Left corner where the text should be drawn
<i>y</i>	The y-Coordinate of the Top-Left corner where the text should be drawn
<i>color</i>	The 16-bit foreground color of the text
<i>bgcolor</i>	The 16-bit background color of the text. You may pass TRANSPARENT as Color
<i>font</i>	The Fontnum to use for drawing
<i>format</i>	The format string (like printf)
<i>...</i>	The arguments to format (like printf)

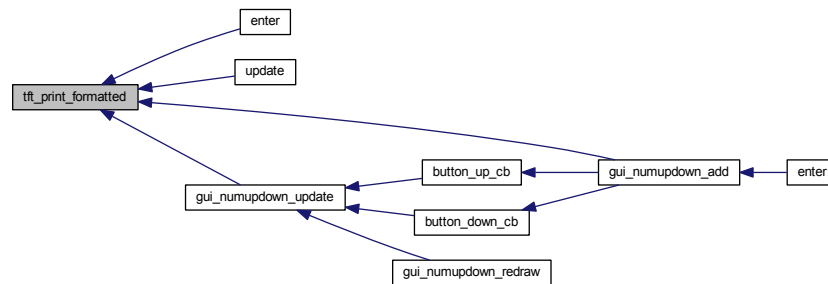
Definition at line 111 of file [tft.c](#).

```
00112 {
00113     static char buffer[128]; //buffer to save the formatted text into
00114
00115     //Since we have variable arguments, we need to forward them. We have to use vsprintf instead of sprintf
    for that.
00116     va_list args;
00117     va_start(args, format); //start the varg-list
00118     vsprintf(buffer, format, args); //let vsprintf render the formatted string
00119     tft_print_line(x, y, color, bgcolor, font, buffer); //print the string as normal text
00120     va_end(args); //end the varg-list
00121 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.25.3.14 void tft_print_line (uint16_t x, uint16_t y, uint16_t color, uint16_t bgcolor, uint8_t font, const char * text)

Prints a unformatted/preformatted string onto the display

Parameters

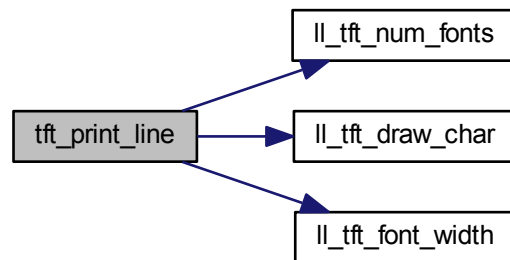
<i>x</i>	The x-Coordinate of the Top-Left corner where the text should be drawn
<i>y</i>	The y-Coordinate of the Top-Left corner where the text should be drawn
<i>color</i>	The 16-bit foreground color of the text
<i>bgcolor</i>	The 16-bit background color of the text. You may pass TRANSPARENT as Color
<i>font</i>	The Fontnum to use for drawing
<i>text</i>	The text to draw

Definition at line 98 of file [tft.c](#).

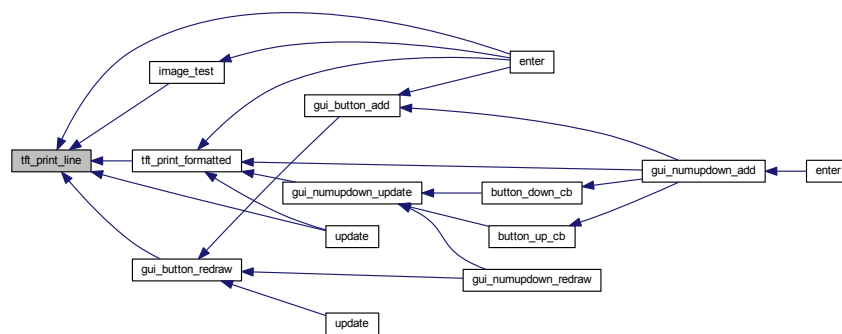
```

00099 {
00100     if (font >= ll_tft_num_fonts()) {
00101         return;    //invalid font index
00102     }
00103
00104     for (int i = 0; i < strlen(text); i++) { //for each char in the line
00105         ll_tft_draw_char(x, y, color, bgcolor, font, text[i]); //draw the char
00106         x += ll_tft_font_width(font); //and increase the x position
00107     }
00108 }
  
```

Here is the call graph for this function:

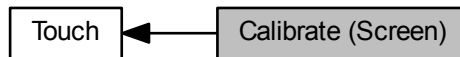


Here is the caller graph for this function:



5.26 Calibrate (Screen)

Collaboration diagram for Calibrate (Screen):



Functions

- [SCREEN_STRUCT * get_screen_calibrate \(\)](#)

5.26.1 Detailed Description

The calibrate screen for the touch module

5.26.2 Function Documentation

5.26.2.1 [SCREEN_STRUCT* get_screen_calibrate \(\)](#)

Returns a pointer to the calibrate screen

See also

[gui_screen_navigate](#)

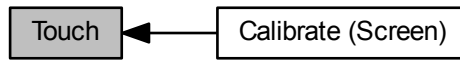
Returns

Definition at line 118 of file [screen_calibrate.c](#).

```
00119 {
00120     return &screen;
00121 }
```

5.27 Touch

Collaboration diagram for Touch:



Modules

- [Calibrate \(Screen\)](#)

Data Structures

- struct [TOUCH_AREA_STRUCT](#)
- struct [POINT_STRUCT](#)

Typedefs

- typedef void(* [TOUCH_CALLBACK](#)) (void *touchArea, [TOUCH_ACTION](#) triggeredAction)

Enumerations

- enum [TOUCH_STATE](#) { [TOUCH_UP](#), [TOUCH_DOWN](#) }
- enum [TOUCH_ACTION](#) {
[NONE](#) = 0x00, [PEN_DOWN](#) = 0x01, [PEN_UP](#) = 0x02, [PEN_ENTER](#) = 0x04,
[PEN_LEAVE](#) = 0x08, [PEN_MOVE](#) = 0x10 }

Functions

- bool [touch_init](#) ()
- bool [touch_add_raw_event](#) (uint16_t x, uint16_t y, [TOUCH_STATE](#) state)
- bool [touch_have_empty](#) (unsigned char num)
- bool [touch_register_area](#) ([TOUCH_AREA_STRUCT](#) *area)
- void [touch_unregister_area](#) ([TOUCH_AREA_STRUCT](#) *area)
- [POINT_STRUCT](#) [touch_get_last_point](#) ()
- void [touch_set_calibration_values](#) (int xs, int dx, int ys, int dy)
- void [touch_set_value_convert_mode](#) (bool [use_calibration](#))

5.27.1 Detailed Description

The Touch module provides access to the touch controller, and executes a callback if a certain region is touched

5.27.2 Typedef Documentation

5.27.2.1 typedef void(* TOUCH_CALLBACK) (void *touchArea, TOUCH_ACTION triggeredAction)

Prototype for Event Listeners (called for every occurring, hooked action)

Note

You should NOT execute long running things in this callback nor should you update the gui. But you can call [gui_screen_navigate\(\)](#) for instance.

Parameters

<i>touchArea</i>	The pointer to the TOUCH_AREA_STRUCT in which the event occurred
<i>triggeredAction</i>	The Action which occurred

Definition at line 67 of file [touch.h](#).

5.27.3 Enumeration Type Documentation

5.27.3.1 enum TOUCH_ACTION

Enum to describe the hooked actions for which you want to receive events for. You can OR-combine them.

See also

[touch_register_area](#)

Enumerator

NONE Do not receive any events.

PEN_DOWN Receive an event when the pen goes down inside the region.

PEN_UP Receive an event when the pen goes up inside the region.

PEN_ENTER Receive an event when the pen enters the region (pen was down before)

PEN_LEAVE Receive an event when the pen leaves the region (pen was inside region before)

PEN_MOVE Receive an event when the pen moves inside the region (pen is down)

Definition at line 52 of file [touch.h](#).

```
00052     {
00053     NONE = 0x00,
00054     PEN_DOWN = 0x01,
00055     PEN_UP = 0x02,
00056     PEN_ENTER = 0x04,
00057     PEN_LEAVE = 0x08,
00058     PEN_MOVE = 0x10
00059 } TOUCH_ACTION;
```

5.27.3.2 enum TOUCH_STATE

Enum to describe the current Touch State.

See also

[touch_add_raw_event](#)

Enumerator

TOUCH_UP The display is currently not touched.

TOUCH_DOWN The display is currently touched at some point.

Definition at line 43 of file [touch.h](#).

```
00043 {
00044     TOUCH_UP,
00045     TOUCH_DOWN
00046 } TOUCH_STATE ;
```

5.27.4 Function Documentation

5.27.4.1 bool touch_add_raw_event (uint16_t x, uint16_t y, TOUCH_STATE state)

Processes a native touch event. Call this function when the pen goes down ([TOUCH_DOWN](#)), when it moves ([TOUCH_DOWN](#)) and also when it goes up again ([TOUCH_UP](#))! It's safe to call this function from an (SPI)-Interrupt.

Parameters

<i>x</i>	The x-Coordinate of the touch event
<i>y</i>	The y-Coordinate of the touch event
<i>state</i>	Whether the pen is up or down

Returns

True on success

Definition at line 72 of file [touch.c](#).

```
00073 {
00074     //Update current and old position/state
00075     bool penDown = (state == TOUCH_DOWN);
00076     bool oldPenDown = (oldState == TOUCH_DOWN);
00077     oldState = state;
00078
00079     if (calibration) { //If in Calibration mode
00080         if (penDown) {
00081             pos.x = touchX;
00082             pos.y = touchY;
00083         } else {
00084             if (oldPenDown) { //Run only if we got at least one pen down
00085                 calibration = 0; //Calibration finish (Touch X and Y are the values from the
last measure, where the pen was down)
00086             }
00087         }
00088     }
00089     return true;
00090 }
00091
00092 //If we reach this point we're not in calibration mode and we need to process the event and call the
registered handlers..
00093
00094 if (use_calibration) { //the underlying touch hardware uses calibration
00095     //Calculate the real touch position out of the passed ones, and the calibration values
00096     pos.x = touchX = (((long)(DWIDTH - 2 * CCENTER) * 2 * (long)((long)touchX -
cal_xs) / cal_dx + 1) >> 1) + CCENTER;
00097     pos.y = touchY = (((long)(DHEIGHT - 2 * CCENTER) * 2 * (long)((long)touchY -
cal_ys) / cal_dy + 1) >> 1) + CCENTER;
00098 } else { //no conversion needed for the underlying hardware
00099     pos.x = touchX;
00100     pos.y = touchY;
00101 }
00102
00103 if (penDown) { //pen is down now
00104     //tft_draw_pixel(touchX,touchY,WHITE);
00105     if (!oldPenDown) { //pen wasn't down before (positive edge) => First Touch
00106         for (int z = 0; z < NUM_AREAS; z++) { // For every touch area
00107             //Check if pos is inside area
00108             if (areas[z] != NULL && touchX >= areas[z]->x1 && touchX <=
areas[z]->x2 && touchY >= areas[z]->y1 && touchY <= areas[z]->y2) {
00109                 areas[z]->flags = 1; //Save PenInside=1
00110             }
00111             if (areas[z]->hookedActions & PEN_DOWN) { //The user wants to receive pen
down events
00112                 areas[z]->callback(areas[z], PEN_DOWN); //Send event to user
```

```

        callback
00113         }
00114     }
00115 }
00116 } else { //Pen was down before => Second, Third event in row
00117     for (int z = 0; z < NUM_AREAS; z++) { // For every touch area
00118         if (areas[z] != NULL) {
00119             //Check if pos is inside area
00120             if (touchX >= areas[z]->x1 && touchX <= areas[z]->x2 && touchY >=
areas[z]->y1 && touchY <= areas[z]->y2) {
00121                 if (areas[z]->flags == 0) { //Pen was not inside before (PenInside==0)
00122                     areas[z]->flags = 1; //Pen is inside now (PenInside=1)
00123                 }
00124                 if (areas[z]->hookedActions & PEN_ENTER) { //The user wants to
receive pen enter events
00125                     areas[z]->callback(areas[z], PEN_ENTER);
00126                 }
00127             }
00128             } else if (areas[z]->flags) { //Pos not inside area, but it was before
(PenInside==1)
00129                 areas[z]->flags = 0; //Pen is no longer inside (PenInside=0)
00130             }
00131             if (areas[z]->hookedActions & PEN_LEAVE) { //The user wants to
receive pen leave events
00132                 areas[z]->callback(areas[z], PEN_LEAVE);
00133             }
00134         }
00135     }
00136 }
00137 }
00138 }
00139 for (int z = 0; z < NUM_AREAS; z++) { // For every touch area
00140     if (areas[z] != NULL && (areas[z]->hookedActions &
PEN_MOVE)) { //User want's to receive pen move events
00141         //Check if pos is inside area
00142         if (touchX >= areas[z]->x1 && touchX <= areas[z]->x2 && touchY >=
areas[z]->y1 && touchY <= areas[z]->y2) {
00143             areas[z]->callback(areas[z], PEN_MOVE);
00144         }
00145     }
00146 }
00147 } else { //pen is not down now
00148     if (oldPenDown) { //but it was down before (negative edge)
00149         for (int z = 0; z < NUM_AREAS; z++) { // For every touch area
00150             //Check if pos is inside area
00151             if (areas[z] != NULL && touchX >= areas[z]->x1 && touchX <=
areas[z]->x2 && touchY >= areas[z]->y1 && touchY <= areas[z]->y2) {
00152                 areas[z]->flags = 0; //The pen is no longer inside (PenInside = 0);
00153             }
00154             if (areas[z]->hookedActions & PEN_UP) { //user want's to receive pen up
events
00155                 areas[z]->callback(areas[z], PEN_UP);
00156             }
00157         }
00158     }
00159 }
00160 }
00161 }
00162 return true;
00163 }

```

5.27.4.2 POINT_STRUCT touch_get_last_point ()

Gets the last touched point

Returns

The Coordinates of the last touched points

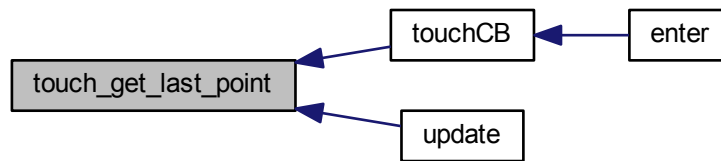
Definition at line 211 of file [touch.c](#).

```

00212 {
00213     return pos;
00214 }

```


Here is the caller graph for this function:



5.27.4.3 bool touch_have_empty (unsigned char num)

Checks whether or not we have memory to manage and track additional `num` [TOUCH_AREA_STRUCTS](#)

Parameters

<code>num</code>	The number of touch areas you would like to allocate
------------------	--

Returns

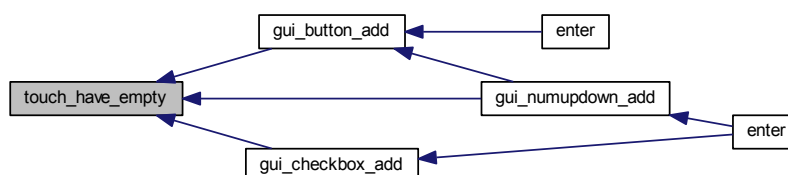
True if there's enough memory to allocate `num` `TOUCH_AREAs`

Definition at line 165 of file [touch.c](#).

```

00166 {
00167     //go through pointer array and check for free spaces
00168     for (unsigned char i = 0; i < NUM_AREAS; i++) {
00169         if (areas[i] == NULL) {
00170             num--;    //a free space was found, we need one less
00171         }
00172     }
00173     if (num == 0) {
00174         return true;    //enough free spaces found
00175     }
00176 }
00177
00178 return false; //not enough free spaces found
00179 }
  
```

Here is the caller graph for this function:



5.27.4.4 bool touch_init ()

Initializes the Touch Controller. Call this method before using any `touch_*` functions

Returns

true on success

Definition at line 61 of file [touch.c](#).

```
00062 {
00063     return ll_touch_init();
00064 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.27.4.5 bool touch_register_area (TOUCH_AREA_STRUCT * area)

Registers a new touch Area. You will receive events for this area from now on.

Parameters

<i>area</i>	A pointer to the configured TOUCH_AREA_STRUCT
-------------	---

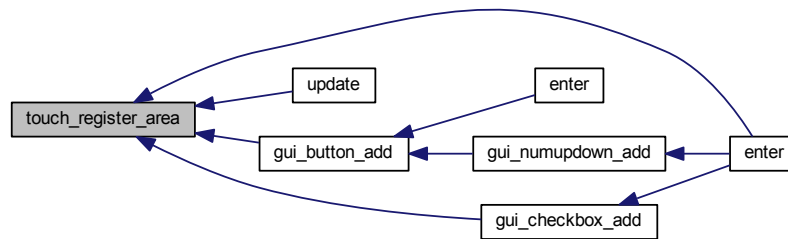
Returns

True if everything was successful and the corresponding Touch Area will be monitored from now on

Definition at line 181 of file [touch.c](#).

```
00182 {
00183     //go through pointer array and check for free space
00184     for (unsigned char i = 0; i < NUM_AREAS; i++) {
00185         if (areas[i] == NULL) { //free space found
00186             area->flags = 0; //we start with empty flags (PenInside=0)
00187             areas[i] = area; //save pointer into list
00188             return true;
00189         }
00190     }
00191     return false; //no free space found
00192 }
00193 }
```

Here is the caller graph for this function:



5.27.4.6 void touch_set_calibration_values (int xs, int dx, int ys, int dy)

Set's the new calibration values

Parameters

xs	x offset (to calibration point 1)
dx	x difference (between calibration point 1 and 2)
ys	y offset (to calibration point 1)
dy	y difference (between calibration point 1 and 2)

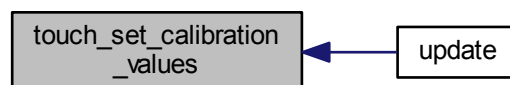
Definition at line 51 of file [touch.c](#).

```

00052 {
00053     cal_xs = xs;
00054     cal_ys = ys;
00055     cal_dx = dx;
00056     cal_dy = dy;
00057 }

```

Here is the caller graph for this function:



5.27.4.7 void touch_set_value_convert_mode (bool use_calibration)

Set's the new value convert mode. Per default use_calibration is false.

Parameters

<i>use_calibration</i>	whether or not the current platform needs display calibration
------------------------	---

Definition at line 66 of file [touch.c](#).

```
00067 {
00068     use_calibration = uc;
00069 }
```

5.27.4.8 void touch_unregister_area (TOUCH_AREA_STRUCT * area)

Unregisters a touch area. You will no longer receive events for this area

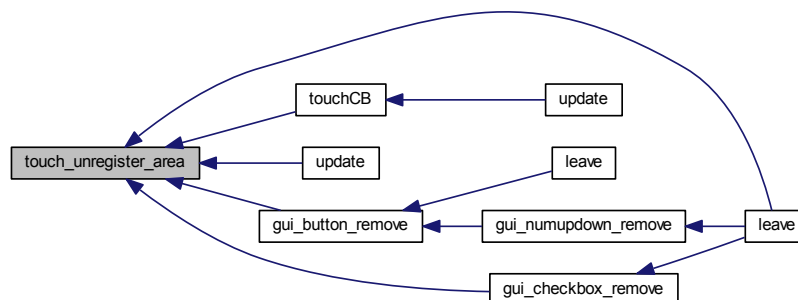
Parameters

<i>area</i>	A pointer to the TOUCH_AREA_STRUCT instance
-------------	---

Definition at line 195 of file [touch.c](#).

```
00196 {
00197     if (area == NULL) {
00198         return;
00199     }
00200     //go through pointer array and find the area to remove
00201     for (unsigned char i = 0; i < NUM_AREAS; i++) {
00202         if (areas[i] == area) { //area found in pointer array at pos i
00203             areas[i] = NULL; //set pointer in list to NULL again
00204             break;
00205         }
00206     }
00207 }
00208 }
```

Here is the caller graph for this function:



Chapter 6

Data Structure Documentation

6.1 Block Struct Reference

```
#include <pixy.h>
```

Data Fields

- uint16_t [type](#)
- uint16_t [signature](#)
- uint16_t [x](#)
- uint16_t [y](#)
- uint16_t [width](#)
- uint16_t [height](#)
- int16_t [angle](#)

6.1.1 Detailed Description

Definition at line [53](#) of file [pixy.h](#).

6.1.2 Field Documentation

6.1.2.1 int16_t angle

Definition at line [84](#) of file [pixy.h](#).

6.1.2.2 uint16_t height

Definition at line [83](#) of file [pixy.h](#).

6.1.2.3 uint16_t signature

Definition at line [79](#) of file [pixy.h](#).

6.1.2.4 uint16_t type

Definition at line [78](#) of file [pixy.h](#).

6.1.2.5 uint16_t width

Definition at line 82 of file [pixy.h](#).

6.1.2.6 uint16_t x

Definition at line 80 of file [pixy.h](#).

6.1.2.7 uint16_t y

Definition at line 81 of file [pixy.h](#).

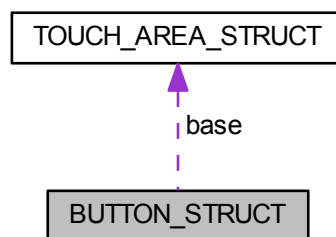
The documentation for this struct was generated from the following file:

- [common/pixy/pixy.h](#)

6.2 BUTTON_STRUCT Struct Reference

```
#include <button.h>
```

Collaboration diagram for BUTTON_STRUCT:



Data Fields

- [TOUCH_AREA_STRUCT base](#)
Basic geometry of the button. You only need to set the x1, y1, x2, y2 members of this struct.
- [uint16_t bgcolor](#)
The 16-bit background color of the button.
- [BUTTON_CALLBACK callback](#)
Callback which is executed when the button is pressed.
- [uint16_t txtcolor](#)
The 16-bit text color.
- [uint8_t font](#)
The number of the font to use.
- [const char * text](#)
The label of the button.

6.2.1 Detailed Description

Structure to configure the Button

Definition at line 55 of file [button.h](#).

6.2.2 Field Documentation

6.2.2.1 TOUCH_AREA_STRUCT base

Basic geometry of the button. You only need to set the x1, y1, x2, y2 members of this struct.

Definition at line 56 of file [button.h](#).

6.2.2.2 uint16_t bgcolor

The 16-bit background color of the button.

Definition at line 57 of file [button.h](#).

6.2.2.3 BUTTON_CALLBACK callback

Callback which is executed when the button is pressed.

Definition at line 58 of file [button.h](#).

6.2.2.4 uint8_t font

The number of the font to use.

Definition at line 60 of file [button.h](#).

6.2.2.5 const char* text

The label of the button.

Definition at line 61 of file [button.h](#).

6.2.2.6 uint16_t txtcolor

The 16-bit text color.

Definition at line 59 of file [button.h](#).

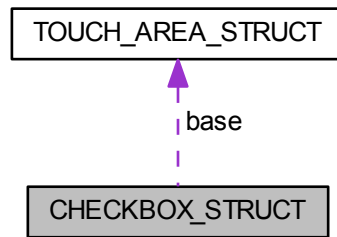
The documentation for this struct was generated from the following file:

- [common/gui/button.h](#)

6.3 CHECKBOX_STRUCT Struct Reference

```
#include <checkbox.h>
```

Collaboration diagram for CHECKBOX_STRUCT:



Data Fields

- [TOUCH_AREA_STRUCT base](#)
Basic geometry of the Checkbox. You only need to set the x1, y1, x2, y2 members of this struct.
- `uint16_t fgcolor`
The 16-bit color of the tickmark.
- `bool checked`
A boolean which indicates whether or not the checkbox is currently checked.
- [CHECKBOX_CALLBACK callback](#)
Callback which is executed when the checkbox changes state.

6.3.1 Detailed Description

Structure to configure the Checkbox

Definition at line 50 of file [checkbox.h](#).

6.3.2 Field Documentation

6.3.2.1 TOUCH_AREA_STRUCT base

Basic geometry of the Checkbox. You only need to set the x1, y1, x2, y2 members of this struct.

Definition at line 51 of file [checkbox.h](#).

6.3.2.2 CHECKBOX_CALLBACK callback

Callback which is executed when the checkbox changes state.

Definition at line 54 of file [checkbox.h](#).

6.3.2.3 bool checked

A boolean which indicates whether or not the checkbox is currently checked.

Definition at line 53 of file [checkbox.h](#).

6.3.2.4 uint16_t fgcolor

The 16-bit color of the tickmark.

Definition at line 52 of file [checkbox.h](#).

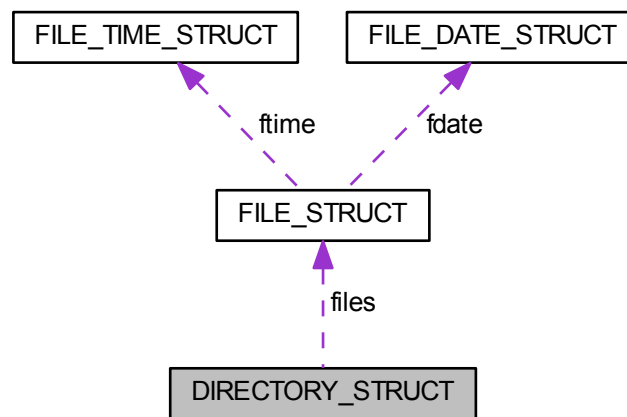
The documentation for this struct was generated from the following file:

- [common/gui/checkbox.h](#)

6.4 DIRECTORY_STRUCT Struct Reference

```
#include <filesystem.h>
```

Collaboration diagram for DIRECTORY_STRUCT:



Data Fields

- `const char * path`
Directory path (absolute)
- `uint16_t num\_files`
Number of files/directories in this directory.
- `FILE\_STRUCT * files`
An array with [num_files](#) [FILE_STRUCT](#) entries.

6.4.1 Detailed Description

Structure which represents an open directory with all it's entries.

See also

[filesystem_dir_open](#)

Definition at line 72 of file [filesystem.h](#).

6.4.2 Field Documentation

6.4.2.1 FILE_STRUCT* files

An array with `num_files` `FILE_STRUCT` entries.

Definition at line 75 of file `filesystem.h`.

6.4.2.2 uint16_t num_files

Number of files/directories in this directory.

Definition at line 74 of file `filesystem.h`.

6.4.2.3 const char* path

Directory path (absolute)

Definition at line 73 of file `filesystem.h`.

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

- `common/filesystem/filesystem.h`

6.5 FILE_DATE_STRUCT Struct Reference

```
#include <filesystem.h>
```

Data Fields

- unsigned `year`: 7
year from 1980 (0..127)
- unsigned `month`: 4
month (1..12)
- unsigned `day`: 5
day (1..31)

6.5.1 Detailed Description

Structure which represents last modified date of a file / directory

Definition at line 43 of file `filesystem.h`.

6.5.2 Field Documentation

6.5.2.1 unsigned day

day (1..31)

Definition at line 46 of file `filesystem.h`.

6.5.2.2 unsigned month

month (1..12)

Definition at line 45 of file [filesystem.h](#).

6.5.2.3 unsigned year

year from 1980 (0..127)

Definition at line 44 of file [filesystem.h](#).

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

- common/filesystem/[filesystem.h](#)

6.6 FILE_HANDLE Struct Reference

```
#include <filesystem.h>
```

Data Fields

- const char * [fname](#)
The absolute file name.
- uint32_t [fpos](#)
The current byte-position in the file.
- uint32_t [fsize](#)
The total file size in bytes.

6.6.1 Detailed Description

Structure which represents an open file.

See also

[filesystem_file_open](#)

Definition at line 81 of file [filesystem.h](#).

6.6.2 Field Documentation

6.6.2.1 const char* fname

The absolute file name.

Definition at line 82 of file [filesystem.h](#).

6.6.2.2 uint32_t fpos

The current byte-position in the file.

See also

[filesystem_file_seek](#)

Definition at line 83 of file [filesystem.h](#).

6.6.2.3 uint32_t fsize

The total file size in bytes.

Definition at line 84 of file [filesystem.h](#).

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

- common/filesystem/[filesystem.h](#)

6.7 FILE_LIST_ENTRY_S Struct Reference

Collaboration diagram for FILE_LIST_ENTRY_S:



Data Fields

- char * [filename](#)
- struct [FILE_LIST_ENTRY_S](#) * [next](#)

6.7.1 Detailed Description

Definition at line 42 of file [screen_photomode_save.c](#).

6.7.2 Field Documentation

6.7.2.1 char* filename

Definition at line 43 of file [screen_photomode_save.c](#).

6.7.2.2 struct FILE_LIST_ENTRY_S* next

Definition at line 44 of file [screen_photomode_save.c](#).

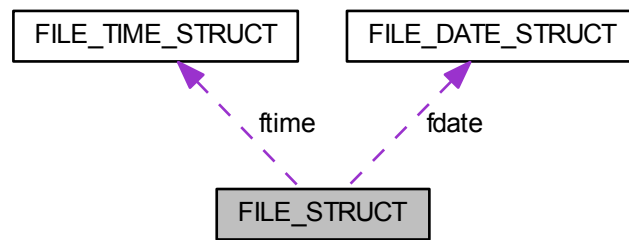
The documentation for this struct was generated from the following file:

- common/app/[screen_photomode_save.c](#)

6.8 FILE_STRUCT Struct Reference

```
#include <filesystem.h>
```

Collaboration diagram for FILE_STRUCT:



Data Fields

- `uint32_t fsize`
File size in bytes. 0 for directories.
- `FILE_DATE_STRUCT fdate`
Last modified date.
- `FILE_TIME_STRUCT ftime`
Last modified time.
- `uint8_t fattrib`
File/Directory Attributes.
- `char * fname`
File/Directory name.

6.8.1 Detailed Description

Structure which represents a file/directory entry.

See also

[DIRECTORY_STRUCT](#)

Definition at line 61 of file [filesystem.h](#).

6.8.2 Field Documentation

6.8.2.1 `uint8_t fattrib`

File/Directory Attributes.

Definition at line 65 of file [filesystem.h](#).

6.8.2.2 `FILE_DATE_STRUCT fdate`

Last modified date.

Definition at line 63 of file [filesystem.h](#).

6.8.2.3 char* fname

File/Directory name.

Definition at line 66 of file [filesystem.h](#).

6.8.2.4 uint32_t fsize

File size in bytes. 0 for directories.

Definition at line 62 of file [filesystem.h](#).

6.8.2.5 FILE_TIME_STRUCT ftime

Last modified time.

Definition at line 64 of file [filesystem.h](#).

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

- common/filesystem/[filesystem.h](#)

6.9 FILE_TIME_STRUCT Struct Reference

```
#include <filesystem.h>
```

Data Fields

- unsigned [hour](#): 5
hour (0..23)
- unsigned [min](#): 6
minute (0..59)
- unsigned [sec](#): 5
second/2 (0..29)

6.9.1 Detailed Description

Structure which represents last modified time of a file / directory

Definition at line 52 of file [filesystem.h](#).

6.9.2 Field Documentation

6.9.2.1 unsigned hour

hour (0..23)

Definition at line 53 of file [filesystem.h](#).

6.9.2.2 unsigned min

minute (0..59)

Definition at line 54 of file [filesystem.h](#).

6.9.2.3 unsigned sec

second/2 (0..29)

Definition at line 55 of file [filesystem.h](#).

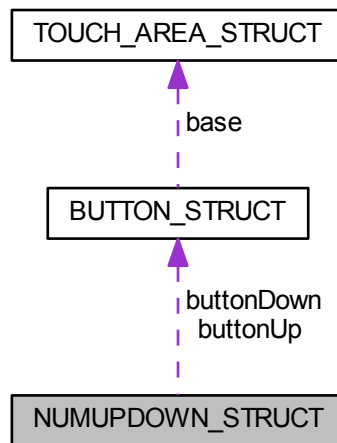
The documentation for this struct was generated from the following file:

- [common/filesystem/filesystem.h](#)

6.10 NUMUPDOWN_STRUCT Struct Reference

```
#include <numupdown.h>
```

Collaboration diagram for NUMUPDOWN_STRUCT:



Data Fields

- [uint16_t x](#)
The x-Coordinate of the Top-Left Starting Point.
- [uint16_t y](#)
The y-Coordinate of the Top-Left Starting Point.
- [uint16_t fgcolor](#)
The 16-bit color of the value-text.
- [int16_t value](#)
The current/default value.
- [int16_t min](#)
The minimum possible value (inclusive)
- [int16_t max](#)
The maximum possible value (inclusive)
- [NUMUPDOWN_CALLBACK callback](#)
Callback which is executed when the value changes.
- [BUTTON_STRUCT buttonUp](#)

For internal use, don't change, don't initialize.

- **BUTTON_STRUCT** `buttonDown`

For internal use, don't change, don't initialize.

6.10.1 Detailed Description

Structure to configure the NumericUpDown

Definition at line 48 of file [numupdown.h](#).

6.10.2 Field Documentation

6.10.2.1 **BUTTON_STRUCT** `buttonDown`

For internal use, don't change, don't initialize.

Definition at line 58 of file [numupdown.h](#).

6.10.2.2 **BUTTON_STRUCT** `buttonUp`

For internal use, don't change, don't initialize.

Definition at line 57 of file [numupdown.h](#).

6.10.2.3 **NUMUPDOWN_CALLBACK** `callback`

Callback which is executed when the value changes.

Definition at line 55 of file [numupdown.h](#).

6.10.2.4 `uint16_t fgcolor`

The 16-bit color of the value-text.

Definition at line 51 of file [numupdown.h](#).

6.10.2.5 `int16_t max`

The maximum possible value (inclusive)

Definition at line 54 of file [numupdown.h](#).

6.10.2.6 `int16_t min`

The minimum possible value (inclusive)

Definition at line 53 of file [numupdown.h](#).

6.10.2.7 `int16_t value`

The current/default value.

Definition at line 52 of file [numupdown.h](#).

6.10.2.8 uint16_t x

The x-Coordinate of the Top-Left Starting Point.

Definition at line 49 of file [numupdown.h](#).

6.10.2.9 uint16_t y

The y-Coordinate of the Top-Left Starting Point.

Definition at line 50 of file [numupdown.h](#).

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

- [common/gui/numupdown.h](#)

6.11 POINT_STRUCT Struct Reference

```
#include <touch.h>
```

Data Fields

- [uint16_t x](#)
The X-Coordinate of the point.
- [uint16_t y](#)
The Y-Coordinate of the point.

6.11.1 Detailed Description

Struct which represents a 2D point on the display

Definition at line 86 of file [touch.h](#).

6.11.2 Field Documentation

6.11.2.1 uint16_t x

The X-Coordinate of the point.

Definition at line 87 of file [touch.h](#).

6.11.2.2 uint16_t y

The Y-Coordinate of the point.

Definition at line 88 of file [touch.h](#).

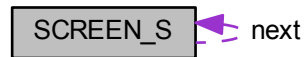
The documentation for this struct was generated from the following file:

- [common/touch/touch.h](#)

6.12 SCREEN_S Struct Reference

```
#include <screen.h>
```

Collaboration diagram for SCREEN_S:



Data Fields

- [SCREEN_CALLBACK on_enter](#)
The Callback which is called when the screen is entered. Add/Register all UI-Elements here.
- [SCREEN_CALLBACK on_leave](#)
The Callback which is called when the screen is left. Remove/Unregister all UI-Elements here.
- [SCREEN_CALLBACK on_update](#)
The Callback which is called repeatedly when the screen should be updated. Update/Redraw all UI-Elements here.
- struct [SCREEN_S](#) * [next](#)
Used internally. do not modify, do not initialize.

6.12.1 Detailed Description

Structure to configure the Screen

Definition at line 52 of file [screen.h](#).

6.12.2 Field Documentation

6.12.2.1 struct SCREEN_S* next

Used internally. do not modify, do not initialize.

Definition at line 57 of file [screen.h](#).

6.12.2.2 SCREEN_CALLBACK on_enter

The Callback which is called when the screen is entered. Add/Register all UI-Elements here.

Definition at line 53 of file [screen.h](#).

6.12.2.3 SCREEN_CALLBACK on_leave

The Callback which is called when the screen is left. Remove/Unregister all UI-Elements here.

Definition at line 54 of file [screen.h](#).

6.12.2.4 SCREEN_CALLBACK on_update

The Callback which is called repeatedly when the screen should be updated. Update/Redraw all UI-Elements here.

Definition at line 55 of file [screen.h](#).

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

- [common/gui/screen.h](#)

6.13 TOUCH_AREA_STRUCT Struct Reference

```
#include <touch.h>
```

Data Fields

- [TOUCH_ACTION hookedActions](#)
Actions to listen to.
- [uint16_t x1](#)
Top Left X-Coordinate of Area.
- [uint16_t y1](#)
Top Left Y-Coordinate of Area.
- [uint16_t x2](#)
Bottom Right X-Coordinate of Area.
- [uint16_t y2](#)
Bottom Right Y-Coordinate of Area.
- [TOUCH_CALLBACK callback](#)
Callback which is executed when an event occurred in this Area.
- [uint8_t flags](#)
For internal use, don't change, don't initialize.

6.13.1 Detailed Description

Structure to configure a Touch Area

Definition at line 72 of file [touch.h](#).

6.13.2 Field Documentation

6.13.2.1 TOUCH_CALLBACK callback

Callback which is executed when an event occurred in this Area.

Definition at line 78 of file [touch.h](#).

6.13.2.2 uint8_t flags

For internal use, don't change, don't initialize.

Definition at line 79 of file [touch.h](#).

6.13.2.3 TOUCH_ACTION hookedActions

Actions to listen to.

Definition at line 73 of file [touch.h](#).

6.13.2.4 uint16_t x1

Top Left X-Coordinate of Area.

Definition at line 74 of file [touch.h](#).

6.13.2.5 uint16_t x2

Bottom Right X-Coordinate of Area.

Definition at line 76 of file [touch.h](#).

6.13.2.6 uint16_t y1

Top Left Y-Coordinate of Area.

Definition at line 75 of file [touch.h](#).

6.13.2.7 uint16_t y2

Bottom Right Y-Coordinate of Area.

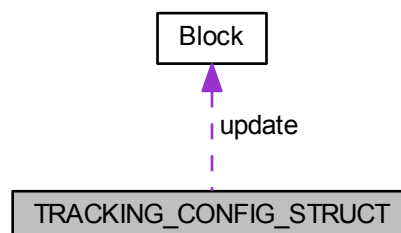
Definition at line 77 of file [touch.h](#).

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

- [common/touch/touch.h](#)

6.14 TRACKING_CONFIG_STRUCT Struct Reference

Collaboration diagram for TRACKING_CONFIG_STRUCT:



Data Fields

- [TRACKING_VOID_CALLBACK start](#)
- [TRACKING_VOID_CALLBACK stop](#)
- [TRACKING_BLOCK_CALLBACK update](#)

6.14.1 Detailed Description

Definition at line [103](#) of file [screen_tracking.c](#).

6.14.2 Field Documentation

6.14.2.1 TRACKING_VOID_CALLBACK start

Definition at line [104](#) of file [screen_tracking.c](#).

6.14.2.2 TRACKING_VOID_CALLBACK stop

Definition at line [105](#) of file [screen_tracking.c](#).

6.14.2.3 TRACKING_BLOCK_CALLBACK update

Definition at line [106](#) of file [screen_tracking.c](#).

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

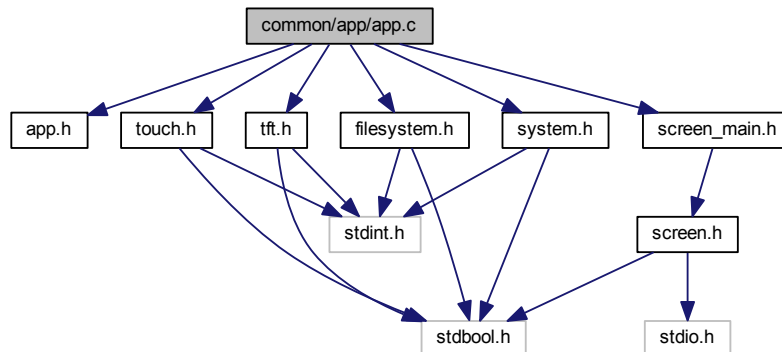
- [common/app/screen_tracking.c](#)

Chapter 7

File Documentation

7.1 common/app/app.c File Reference

```
#include "app.h"
#include "tft.h"
#include "system.h"
#include "touch.h"
#include "screen_main.h"
#include "filesystem.h"
Include dependency graph for app.c:
```



Functions

- void `app_init()`
- void `app_process()`

7.2 app.c

```
00001 /*****
00002  * Project:      discoverpixy
00003  * Website:     https://github.com/t-moe/discoverpixy
00004  * Authors:     Aaron Schmocker, Timo Lang
00005  * Institution:  BFH Bern University of Applied Sciences
00006 *****/
```

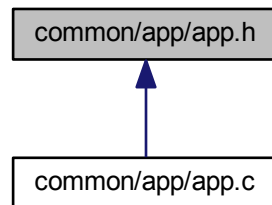
```

00006 * File:                common/app/app.c
00007 *
00008 * Version History:
00009 * Date      Autor Email      SHA      Changes
00010 * 2015-04-03 timolang@gmail.com 51089aa Refactored Project Structure for use with emulator
00011 * 2015-04-03 timolang@gmail.com 1f2af9f Added more tft functions to common and emulator. Fixed
eventloop.
00012 * 2015-04-03 timolang@gmail.com cab8609 Integrated pixy into emulator. Pixy is no longer in the common/
libs folder but in emulator/libs and discovery/libs
00013 * 2015-04-03 timolang@gmail.com 1aa9194 Fixed Drawing of rects in Emulator. Got frames from pixy to
emulator. Sloooooow.
00014 * 2015-04-25 timolang@gmail.com 416883c Pixy&Usb work with the new folder structure now.
00015 * 2015-04-25 timolang@gmail.com 3d1e4b2 Simplified code a bit. Emulator does not work stable when
replugging pixy.
00016 * 2015-04-25 timolang@gmail.com 0858b0d Fixed some bugs when receiving large data.
00017 * 2015-04-27 timolang@gmail.com 259d446 Added touch support to emulator. Implemented basic touch
function.
00018 * 2015-04-27 aaron@duckpond.ch aed90ef Drawcircle added (emulator)
00019 * 2015-04-27 timolang@gmail.com e249fb2 Added font support
00020 * 2015-04-27 timolang@gmail.com 7c9eabc Added button support.
00021 * 2015-04-27 timolang@gmail.com b300ac5 Added Checkbox support
00022 * 2015-04-27 timolang@gmail.com cf72baa Introduced a Screen (sub) module and divided app into multiple
screens.
00023 * 2015-05-10 timolang@gmail.com e2bce8f Added filesystem module, tests and implementation for it in
emulator.
00024 * 2015-05-12 aaron@duckpond.ch aec62d4 Added datasheets, updated touchsupport.
00025 * 2015-05-28 aaron@duckpond.ch 5bda699 implemented functions to get x and y coordinates and a test
function
00026 * 2015-05-29 aaron@duckpond.ch 7d2d1a1 Implemented basic sampling and averaging of touch coordinates
using timer7
00027 *
00028 *****
*****/
00029
00030 #include "app.h"
00031 #include "tft.h"
00032 #include "system.h"
00033 #include "touch.h"
00034 #include "screen_main.h"
00035 #include "filesystem.h"
00036
00037 void app_init()
00038 {
00039     system_init();
00040     tft_init();
00041     touch_init();
00042     filesystem_init();
00043
00044     gui_screen_navigate(get_screen_main());
00045 }
00046
00047 //app event loop
00048 void app_process()
00049 {
00050
00051     system_process(); //Let the system handle it's pending events
00052     gui_screen_update(); //update the currently active screen
00053 }

```


7.3 common/app/app.h File Reference

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



Functions

- void `app_init()`
- void `app_process()`

7.4 app.h

```

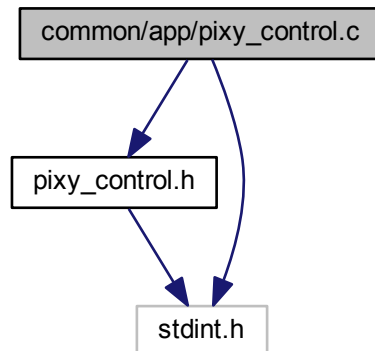
00001 /*****
00002  * Project:      discoverpixy
00003  * Website:     https://github.com/t-moe/discoverpixy
00004  * Authors:     Aaron Schmocker, Timo Lang
00005  * Institution:  BFH Bern University of Applied Sciences
00006  * File:        common/app/app.h
00007  *
00008  * Version History:
00009  * Date          Autor Email          SHA      Changes
00010  * 2015-04-03    timolang@gmail.com    51089aa  Refactored Project Structure for use with emulator
00011  * 2015-05-10    timolang@gmail.com    21edc56  Added doxyfile (doxygen) for the common folder. Started with
00012  * 2015-05-11    timolang@gmail.com    08d9fe0  More work on doxygen module structure
00013  * 2015-06-08    timolang@gmail.com    73db8b5  Added doxygen mainpage comment in app.h
00014  *
00015  *****/
00016
00017 #ifndef APP_H
00018 #define APP_H
00019
00020
00021
00022 void app_init();
00023
00024 void app_process();
00025
00026 #endif /* APP_H */
  
```

7.5 common/app/pixy_control.c File Reference

```

#include <pixy_control.h>
#include <stdint.h>
  
```

Include dependency graph for `pixy_control.c`:



Macros

- `#define REG_PID_KP` (0.5f)
- `#define REG_PID_KI` (0.001f)
- `#define REG_PID_KD` (0.001f)
- `#define REG_PID_TA` (0.01f)

Functions

- `int16_t pixy_PID_Y` (`int16_t x`, `int16_t w`)
- `int16_t pixy_PID_X` (`int16_t x`, `int16_t w`)

7.5.1 Macro Definition Documentation

7.5.1.1 `#define REG_PID_KD` (0.001f)

Definition at line 39 of file `pixy_control.c`.

7.5.1.2 `#define REG_PID_KI` (0.001f)

Definition at line 38 of file `pixy_control.c`.

7.5.1.3 `#define REG_PID_KP` (0.5f)

Definition at line 37 of file `pixy_control.c`.

7.5.1.4 `#define REG_PID_TA` (0.01f)

Definition at line 40 of file `pixy_control.c`.

7.6 pixy_control.c

```

00001 /*****
00002 * Project:      discoverpixy
00003 * Website:     https://github.com/t-moe/discoverpixy
00004 * Authors:     Aaron Schmocker, Timo Lang
00005 * Institution:  BFH Bern University of Applied Sciences
00006 * File:        common/app/pixy_control.c
00007 *
00008 * Version History:
00009 * Date      Autor Email      SHA      Changes
00010 * 2015-06-02 aaron@duckpond.ch e018a75 Implemented basic pi and pid controller
00011 * 2015-06-06 aaron@duckpond.ch 8c264c2 Comment refactoring, updated PID values
00012 * 2015-06-06 aaron@duckpond.ch a04cda9 Refactured comments and implemented a bugfix for the PID
00013 *           controller
00014 * 2015-06-07 aaron@duckpond.ch 802d3df Fixed pid controller and refactored code
00015 * 2015-06-07 aaron@duckpond.ch 3d98ca9 Minor changes
00016 * 2015-06-07 timolang@gmail.com c87220d Renamed pixy_helper to pixy_frame. Updated docu of appliaction.
00017 *           added doxygen comments to pixy_{frame,control}.h
00018 *
00019 *****/
00020 /*
00021 * pixy_control.c
00022 *
00023 * Notation
00024 * -----
00025 * x      : Sollwert (Führgrösse)
00026 * w      : Istwert (Reglergrösse)
00027 * esum   : Integralteil
00028 * e      : Regelabweichung
00029 * y      : Stellgrösse
00030 *
00031 */
00032 */
00033 #include<pixy_control.h>
00034 #include<stdint.h>
00035
00036 // PID tuning factors
00037 #define REG_PID_KP      (0.5f)
00038 #define REG_PID_KI      (0.001f)
00039 #define REG_PID_KD      (0.001f)
00040 #define REG_PID_TA      (0.01f)
00041
00042
00043 // PID controller implementatoin for the y-axis
00044 int16_t pixy_PID_Y(int16_t x, int16_t w)
00045 {
00046     float e = 0;
00047     static float esum = 0;
00048     static float eold = 0;
00049     float y = 0;
00050
00051     e = (float)(x - w); // calculate the controller offset
00052
00053     //----PID-control-----
00054     esum = esum + e; // add e to the current sum
00055
00056     y += REG_PID_KP * e; // add the proportional part to the output
00057     y += REG_PID_KI * REG_PID_TA * esum; // add the integral part to the output
00058     y += REG_PID_KD * (e - eold) / REG_PID_TA; // add the differential part to the
00059     output
00060     //-----
00061     eold = e; // save the previous value
00062
00063     return (int16_t)y;
00064 }
00065
00066 // PID controller implementation for the x-axis
00067 int16_t pixy_PID_X(int16_t x, int16_t w)
00068 {
00069     float e = 0;
00070     static float esum = 0;
00071     static float eold = 0;
00072     float y = 0;
00073
00074     e = (float)(x - w); // calculate the controller offset
00075
00076     //----PID-control-----
00077     esum = esum + e; // add e to the current sum
00078
00079     y += REG_PID_KP * e; // add the proportional part to the output

```

```

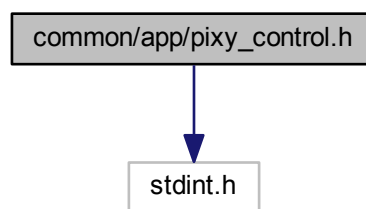
00080     y += REG_PID_KI * REG_PID_TA * esum;           // add the integral part to the output
00081     y += REG_PID_KD * (e - eold) / REG_PID_TA;      // add the differential part to the
output
00082     //-----
00083
00084     eold = e;                                       // save the previous value
00085
00086     return (int16_t)y;
00087 }

```

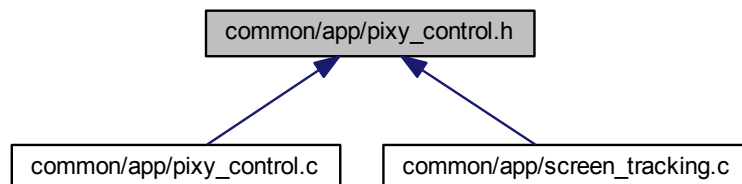
7.7 common/app/pixy_control.h File Reference

```
#include <stdint.h>
```

Include dependency graph for pixy_control.h:



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



Functions

- `int16_t pixy_PID_Y(int16_t x, int16_t w)`
- `int16_t pixy_PID_X(int16_t x, int16_t w)`

7.8 pixy_control.h

```

00001 /*****
00002  * Project:      discoverpixy
00003  * Website:     https://github.com/t-moe/discoverpixy
00004  * Authors:     Aaron Schmocker, Timo Lang

```

```

00005 * Institution:    BFH Bern University of Applied Sciences
00006 * File:          common/app/pixy_control.h
00007 *
00008 * Version History:
00009 * Date           Autor Email      SHA      Changes
00010 * 2015-06-02     aaron@duckpond.ch e018a75 Implemented basic pi and pid controller
00011 * 2015-06-06     aaron@duckpond.ch 8c264c2 Comment refactoring, updated PID values
00012 * 2015-06-06     aaron@duckpond.ch a04cda9 Refactored comments and implemented a bugfix for the PID
00013 * 2015-06-07     aaron@duckpond.ch 802d3df Fixed pid controller and refactored code
00014 * 2015-06-07     timolang@gmail.com c87220d Renamed pixy_helper to pixy_frame. Updated docu of appliaction.
00015 * added doxygen comments to pixy_{frame,control}.h
00016 *****
00017
00018 #ifndef PIXY_CONTROL_H_
00019 #define PIXY_CONTROL_H_
00020
00021 #include<stdint.h>
00022
00027
00033
00034
00039
00046 int16_t pixy_PID_Y(int16_t x, int16_t w);
00047
00054 int16_t pixy_PID_X(int16_t x, int16_t w);
00055
00058 #endif /* PIXY_CONTROL_H_ */

```

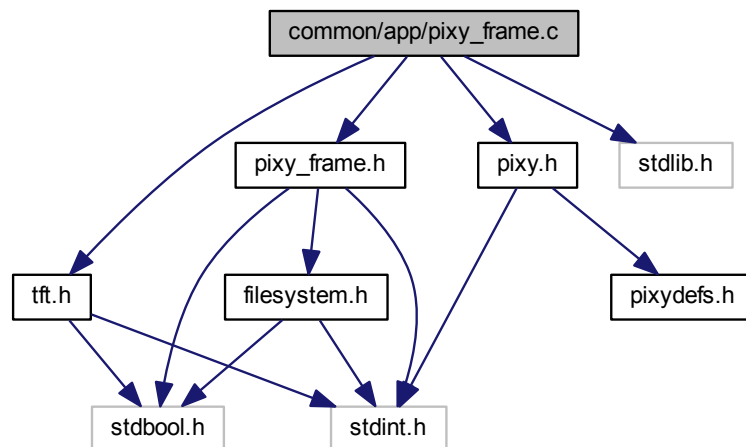
7.9 common/app/pixy_frame.c File Reference

```

#include "pixy_frame.h"
#include "pixy.h"
#include "tft.h"
#include <stdlib.h>

```

Include dependency graph for pixy_frame.c:



Functions

- static int [renderBA81](#) (uint16_t xpos, uint16_t ypos, uint16_t width, uint16_t height, uint32_t frameLen, uint8_t *frame)

- static int [saveBA81](#) ([FILE_HANDLE](#) *handle, uint16_t width, uint16_t height, uint32_t frameLen, uint8_t *frame)
- int [pixy_render_full_frame](#) (uint16_t x, uint16_t y)
- int [pixy_render_cropped_frame](#) (uint16_t x, uint16_t y, uint16_t xoffset, uint16_t yoffset, uint16_t width, uint16_t height)
- int [pixy_save_full_frame](#) ([FILE_HANDLE](#) *handle)
- int [pixy_save_cropped_frame](#) ([FILE_HANDLE](#) *handle, uint16_t xoffset, uint16_t yoffset, uint16_t width, uint16_t height)
- static void [interpolateBayer](#) (uint16_t width, uint16_t x, uint16_t y, uint8_t *pixel, uint8_t *r, uint8_t *g, uint8_t *b)
- int [pixy_cc_set_region](#) (uint8_t signum, uint16_t xoffset, uint16_t yoffset, uint16_t width, uint16_t height)

7.9.1 Function Documentation

7.9.1.1 static void [interpolateBayer](#) (uint16_t *width*, uint16_t *x*, uint16_t *y*, uint8_t * *pixel*, uint8_t * *r*, uint8_t * *g*, uint8_t * *b*) [static]

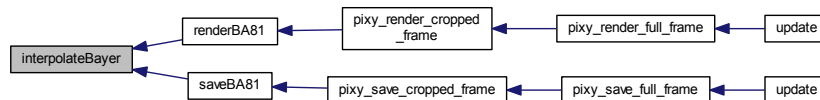
Definition at line 108 of file [pixy_frame.c](#).

```

00109 {
00110     if (y & 1) {
00111         if (x & 1) {
00112             *r = *pixel;
00113             *g = (*(pixel - 1) + * (pixel + 1) + * (pixel + width) + * (pixel - width)) >> 2;
00114             *b = (*(pixel - width - 1) + * (pixel - width + 1) + * (pixel + width - 1) + * (pixel + width +
00115 1)) >> 2;
00116         } else {
00117             *r = (*(pixel - 1) + * (pixel + 1)) >> 1;
00118             *g = *pixel;
00119             *b = (*(pixel - width) + * (pixel + width)) >> 1;
00120         } else {
00121             if (x & 1) {
00122                 *r = (*(pixel - width) + * (pixel + width)) >> 1;
00123                 *g = *pixel;
00124                 *b = (*(pixel - 1) + * (pixel + 1)) >> 1;
00125             } else {
00126                 *r = (*(pixel - width - 1) + * (pixel - width + 1) + * (pixel + width - 1) + * (pixel + width +
00127 1)) >> 2;
00128                 *g = (*(pixel - 1) + * (pixel + 1) + * (pixel + width) + * (pixel - width)) >> 2;
00129                 *b = *pixel;
00130             }
00131         }
00132     }

```

Here is the caller graph for this function:



7.9.1.2 static int [renderBA81](#) (uint16_t *xpos*, uint16_t *ypos*, uint16_t *width*, uint16_t *height*, uint32_t *frameLen*, uint8_t * *frame*) [static]

Definition at line 139 of file [pixy_frame.c](#).

```

00140 {
00141     uint16_t x, y;

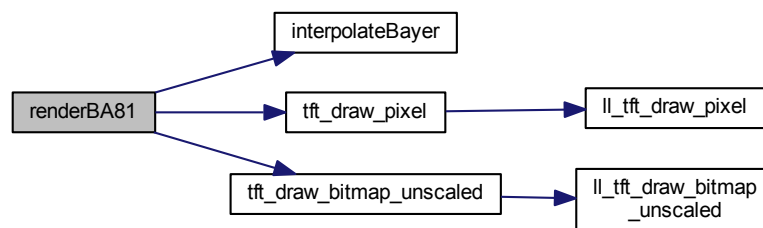
```

```

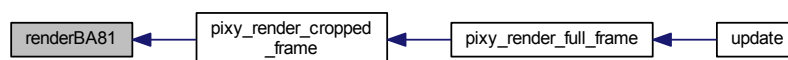
00142     uint8_t r, g, b;
00143
00144
00145     // skip first line
00146     frame += width;
00147
00148     // don't render top and bottom rows, and left and rightmost columns because of color
00149     // interpolation
00150     //uint32_t decodedimage[(width-2)*(height-2)];
00151     uint16_t* decodedimage = malloc(sizeof(uint16_t) * (width - 2) * (height - 2));
00152
00153     if (decodedimage == NULL) { //not enough free space to decode image in memory
00154         //decode & render image pixel by pixel
00155         for (y = 1; y < height - 1; y++) {
00156             frame++;
00157
00158             for (x = 1; x < width - 1; x++, frame++) {
00159                 interpolateBayer(width, x, y, frame, &r, &g, &b);
00160                 tft_draw_pixel(xpos + x - 1, ypos + y - 1, RGB(r, g, b));
00161             }
00162             frame++;
00163         }
00164     } else { //enough space
00165         uint16_t* line = decodedimage;
00166
00167         for (y = 1; y < height - 1; y++) {
00168             //line = (unsigned int *)img.scanLine(y-1);
00169             frame++;
00170
00171             for (x = 1; x < width - 1; x++, frame++) {
00172                 interpolateBayer(width, x, y, frame, &r, &g, &b);
00173                 /*line++ = (0xff<<24) | (r<<16) | (g<<8) | (b<<0);
00174                 *line++ = RGB(r, g, b);
00175             }
00176             frame++;
00177         }
00178     }
00179 }
00180
00181 tft_draw_bitmap_unscaled(xpos, ypos, width - 2, height - 2, decodedimage);
00182
00183 free(decodedimage);
00184 }
00185
00186 return 0;
00187 }

```

Here is the call graph for this function:



Here is the caller graph for this function:



7.9.1.3 `static int saveBA81 ( FILE_HANDLE * handle, uint16_t width, uint16_t height, uint32_t frameLen, uint8_t * frame )`
`[static]`

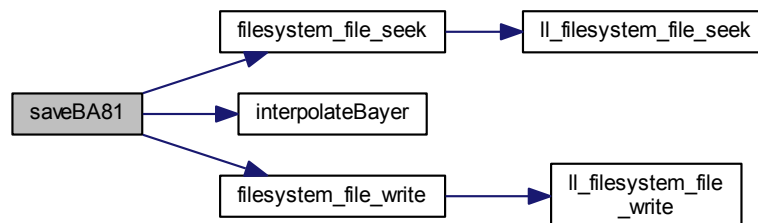
Definition at line 189 of file [pixy_frame.c](#).

```

00190 {
00191     uint16_t x, y;
00192     uint8_t r, g, b;
00193
00194     uint32_t fpos = handle->fpos;
00195     uint32_t row_size_padded = ((width - 2) * 3 + 3) & (~3); //row size aligned to 4 bytes
00196     uint32_t fpos_end = fpos + row_size_padded * (height - 2);
00197
00198
00199     // skip first line
00200     frame += width;
00201
00202     // don't render top and bottom rows, and left and rightmost columns because of color
00203     // interpolation
00204
00205     for (y = 1; y < height - 1; y++) {
00206         frame++;
00207         uint8_t rowbuf[row_size_padded];
00208
00209         //Bitmaps are saved "bottom-up". Seek to the right row.
00210         if (filesystem_file_seek(handle, fpos_end - row_size_padded * y) !=
F_OK) {
00211             return -1;
00212         }
00213
00214         for (x = 1; x < width - 1; x++, frame++) {
00215             interpolateBayer(width, x, y, frame, &r, &g, &b);
00216             //bitmaps are saved in 24bit b,g,r format
00217             rowbuf[(x - 1) * 3] = b;
00218             rowbuf[(x - 1) * 3 + 1] = g;
00219             rowbuf[(x - 1) * 3 + 2] = r;
00220         }
00221
00222         if (filesystem_file_write(handle, rowbuf, row_size_padded) !=
F_OK) {
00223             return -1;
00224         }
00225         frame++;
00226     }
00227
00228     return 0;
00229 }
00230

```

Here is the call graph for this function:



Here is the caller graph for this function:



7.10 pixy_frame.c

```

00001 /*****
00002 * Project:      discoverpixy
00003 * Website:     https://github.com/t-moe/discoverpixy
00004 * Authors:     Aaron Schmocker, Timo Lang
00005 * Institution: BFH Bern University of Applied Sciences
00006 * File:        common/app/pixy_frame.c
00007 *
00008 * Version History:
00009 * Date      Autor Email      SHA      Changes
00010 * 2015-06-07 timolang@gmail.com c87220d Renamed pixy_helper to pixy_frame. Updated docu of appliaction.
00011 * added doxygen comments to pixy_{frame,control}.h
00012 *****/
00013
00014 #include "pixy_frame.h"
00015 #include "pixy.h"
00016 #include "tft.h"
00017 #include <stdlib.h>
00018
00019
00020
00021 static int renderBA81(uint16_t xpos, uint16_t ypos, uint16_t width, uint16_t height, uint32_t
frameLen, uint8_t* frame);
00022 static int saveBA81(FILE_HANDLE* handle, uint16_t width, uint16_t height, uint32_t
frameLen, uint8_t* frame);
00023
00024
00025 int pixy_render_full_frame(uint16_t x, uint16_t y)
00026 {
00027     return pixy_render_cropped_frame(x, y, 0, 0, 320, 200);
00028 }
00029
00030
00031 int pixy_render_cropped_frame(uint16_t x, uint16_t y, uint16_t xoffset, uint16_t
yoffset, uint16_t width, uint16_t height)
00032 {
00033     uint8_t* videodata;
00034     int32_t response;
00035     int32_t fourccc;
00036     int8_t renderflags;
00037     uint16_t xwidth;
00038     uint16_t ywidth;
00039     uint32_t size;
00040
00041
00042     int return_value = pixy_command("cam_getFrame", // String id for remote procedure
00043                                     INT8(0x21), // mode
00044                                     INT16(xoffset), // xoffset
00045                                     INT16(yoffset), // yoffset
00046                                     INT16(width), // width
00047                                     INT16(height), // height
00048                                     END_OUT_ARGS, // separator
00049                                     &response, // pointer to mem address for return value
00050                                     &fourccc,
00051                                     &renderflags,
00052                                     &xwidth,
00053                                     &ywidth,
00054                                     &size,
00055                                     &videodata, // pointer to mem address for returned frame
00056                                     END_IN_ARGS);
00057
00058     if (return_value == 0) {
00059         return_value = renderBA81(x, y, xwidth, ywidth, size, videodata);
00060     }
00061
00062     return return_value;

```

```

00063 }
00064
00065 int pixy_save_full_frame(FILE_HANDLE* handle)
00066 {
00067     return pixy_save_cropped_frame(handle, 0, 0, 320, 200);
00068 }
00069
00070 int pixy_save_cropped_frame(FILE_HANDLE* handle, uint16_t xoffset,
    uint16_t yoffset, uint16_t width, uint16_t height)
00071 {
00072     uint8_t* videodata;
00073     int32_t response;
00074     int32_t fourccc;
00075     int8_t renderflags;
00076     uint16_t xwidth;
00077     uint16_t ywidth;
00078     uint32_t size;
00079
00080
00081     int return_value = pixy_command("cam_getFrame", // String id for remote procedure
00082                                     INT8(0x21), // mode
00083                                     INT16(xoffset), // xoffset
00084                                     INT16(yoffset), // yoffset
00085                                     INT16(width), // width
00086                                     INT16(height), // height
00087                                     END_OUT_ARGS, // separator
00088                                     &response, // pointer to mem address for return value
00089                                     &fourccc,
00090                                     &renderflags,
00091                                     &xwidth,
00092                                     &ywidth,
00093                                     &size,
00094                                     &videodata, // pointer to mem address for returned frame
00095                                     END_IN_ARGS);
00096
00097     if (return_value == 0) {
00098         return_value = saveBA81(handle, xwidth, ywidth, size, videodata);
00099     }
00100
00101     return return_value;
00102 }
00103
00104
00105
00106
00107
00108 static void interpolateBayer(uint16_t width, uint16_t x, uint16_t y, uint8_t* pixel,
    uint8_t* r, uint8_t* g, uint8_t* b)
00109 {
00110     if (y & 1) {
00111         if (x & 1) {
00112             *r = *pixel;
00113             *g = (* (pixel - 1) + * (pixel + 1) + * (pixel + width) + * (pixel - width)) >> 2;
00114             *b = (* (pixel - width - 1) + * (pixel - width + 1) + * (pixel + width - 1) + * (pixel + width +
00115 1)) >> 2;
00116         } else {
00117             *r = (* (pixel - 1) + * (pixel + 1)) >> 1;
00118             *g = *pixel;
00119             *b = (* (pixel - width) + * (pixel + width)) >> 1;
00120         }
00121     } else {
00122         if (x & 1) {
00123             *r = (* (pixel - width) + * (pixel + width)) >> 1;
00124             *g = *pixel;
00125             *b = (* (pixel - 1) + * (pixel + 1)) >> 1;
00126         } else {
00127             *r = (* (pixel - width - 1) + * (pixel - width + 1) + * (pixel + width - 1) + * (pixel + width +
00128 1)) >> 2;
00129             *g = (* (pixel - 1) + * (pixel + 1) + * (pixel + width) + * (pixel - width)) >> 2;
00130             *b = *pixel;
00131         }
00132     }
00133 }
00134
00135
00136
00137
00138
00139 static int renderBA81(uint16_t xpos, uint16_t ypos, uint16_t width, uint16_t height, uint32_t
    frameLen, uint8_t* frame)
00140 {
00141     uint16_t x, y;
00142     uint8_t r, g, b;
00143
00144

```

```

00145     // skip first line
00146     frame += width;
00147
00148     // don't render top and bottom rows, and left and rightmost columns because of color
00149     // interpolation
00150     //uint32_t decodedimage[(width-2)*(height-2)];
00151     uint16_t* decodedimage = malloc(sizeof(uint16_t) * (width - 2) * (height - 2));
00152
00153     if (decodedimage == NULL) { //not enough free space to decode image in memory
00154         //decode & render image pixel by pixel
00155         for (y = 1; y < height - 1; y++) {
00156             frame++;
00157
00158             for (x = 1; x < width - 1; x++, frame++) {
00159                 interpolateBayer(width, x, y, frame, &r, &g, &b);
00160                 tft_draw_pixel(xpos + x - 1, ypos + y - 1, RGB(r, g, b));
00161             }
00162
00163             frame++;
00164         }
00165     } else { //enough space
00166         uint16_t* line = decodedimage;
00167
00168         for (y = 1; y < height - 1; y++) {
00169             //line = (unsigned int *)img.scanLine(y-1);
00170             frame++;
00171
00172             for (x = 1; x < width - 1; x++, frame++) {
00173                 interpolateBayer(width, x, y, frame, &r, &g, &b);
00174                 //*line++ = (0xff<<24) | (r<<16) | (g<<8) | (b<<0);
00175                 *line++ = RGB(r, g, b);
00176             }
00177
00178             frame++;
00179         }
00180
00181         tft_draw_bitmap_unscaled(xpos, ypos, width - 2, height - 2, decodedimage);
00182
00183         free(decodedimage);
00184     }
00185
00186     return 0;
00187 }
00188
00189 static int saveBA81(FILE_HANDLE* handle, uint16_t width, uint16_t height, uint32_t
frameLen, uint8_t* frame)
00190 {
00191     uint16_t x, y;
00192     uint8_t r, g, b;
00193
00194     uint32_t fpos = handle->fpos;
00195     uint32_t row_size_padded = ((width - 2) * 3 + 3) & (~3); //row size aligned to 4 bytes
00196     uint32_t fpos_end = fpos + row_size_padded * (height - 2);
00197
00198     // skip first line
00199     frame += width;
00200
00201     // don't render top and bottom rows, and left and rightmost columns because of color
00202     // interpolation
00203
00204     for (y = 1; y < height - 1; y++) {
00205         frame++;
00206         uint8_t rowbuf[row_size_padded];
00207
00208         //Bitmaps are saved "bottom-up". Seek to the right row.
00209         if (filesystem_file_seek(handle, fpos_end - row_size_padded * y) !=
F_OK) {
00210             return -1;
00211         }
00212
00213         for (x = 1; x < width - 1; x++, frame++) {
00214             interpolateBayer(width, x, y, frame, &r, &g, &b);
00215             //bitmaps are saved in 24bit b,g,r format
00216             rowbuf[(x - 1) * 3] = b;
00217             rowbuf[(x - 1) * 3 + 1] = g;
00218             rowbuf[(x - 1) * 3 + 2] = r;
00219         }
00220
00221         if (filesystem_file_write(handle, rowbuf, row_size_padded) !=
F_OK) {
00222             return -1;
00223         }
00224
00225         frame++;
00226     }
00227 }
00228

```

```

00229     return 0;
00230 }
00231
00232
00233 int pixy_cc_set_region(uint8_t signum, uint16_t xoffset, uint16_t yoffset, uint16_t width
, uint16_t height)
00234 {
00235     int32_t response;
00236
00237     int return_value = pixy_command("cc_setSigRegion", // String id for remote procedure
00238                                     INT32(0), // type = normal color code
00239                                     INT8(signum),
00240                                     INT16(xoffset), // xoffset
00241                                     INT16(yoffset), // yoffset
00242                                     INT16(width), // width
00243                                     INT16(height), // height
00244                                     END_OUT_ARGS, // separator
00245                                     &response, // pointer to mem address for return value
00246                                     END_IN_ARGS);
00247     return return_value;
00248 }

```

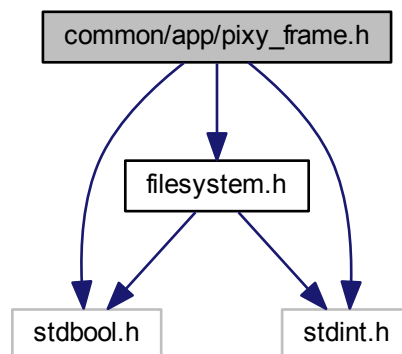
7.11 common/app/pixy_frame.h File Reference

```

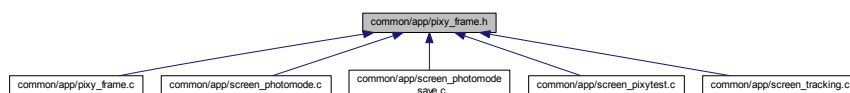
#include <stdbool.h>
#include <stdint.h>
#include "filesystem.h"

```

Include dependency graph for pixy_frame.h:



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



Functions

- int `pixy_render_full_frame` (uint16_t x, uint16_t y)

- `int pixy_render_cropped_frame` (`uint16_t x`, `uint16_t y`, `uint16_t xoffset`, `uint16_t yoffset`, `uint16_t width`, `uint16_t height`)
- `int pixy_save_full_frame` (`FILE_HANDLE *handle`)
- `int pixy_save_cropped_frame` (`FILE_HANDLE *handle`, `uint16_t xoffset`, `uint16_t yoffset`, `uint16_t width`, `uint16_t height`)
- `int pixy_cc_set_region` (`uint8_t signum`, `uint16_t xoffset`, `uint16_t yoffset`, `uint16_t width`, `uint16_t height`)

7.12 pixy_frame.h

```

00001 /*****
00002  * Project:      discoverpixy
00003  * Website:     https://github.com/t-moe/discoverpixy
00004  * Authors:     Aaron Schmocker, Timo Lang
00005  * Institution:  BFH Bern University of Applied Sciences
00006  * File:        common/app/pixy_frame.h
00007  *
00008  * Version History:
00009  * Date      Autor Email      SHA      Changes
00010  * 2015-06-07 timolang@gmail.com c87220d Renamed pixy_helper to pixy_frame. Updated docu of appliacion.
00011  *          added doxygen comments to pixy_{frame,control}.h
00012  * *****/
00013
00014 #ifndef PIXY_FRAME_H
00015 #define PIXY_FRAME_H
00016
00017 #include <stdbool.h>
00018 #include <stdint.h>
00019 #include "filesystem.h"
00020
00021
00026
00033
00034
00039
00040
00047 int pixy_render_full_frame(uint16_t x, uint16_t y);
00048
00059 int pixy_render_cropped_frame(uint16_t x, uint16_t y, uint16_t xoffset, uint16_t
yoffset, uint16_t width, uint16_t height);
00060
00068 int pixy_save_full_frame(FILE_HANDLE* handle);
00069
00079 int pixy_save_cropped_frame(FILE_HANDLE* handle, uint16_t xoffset,
uint16_t yoffset, uint16_t width, uint16_t height);
00080
00090 int pixy_cc_set_region(uint8_t signum, uint16_t xoffset, uint16_t yoffset, uint16_t width
, uint16_t height);
00091
00094 #endif /* PIXY_FRAME_H */

```

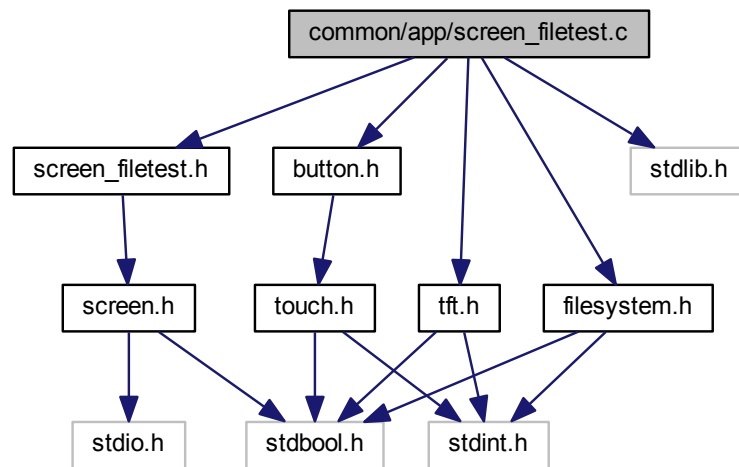
7.13 common/app/screen_filetest.c File Reference

```

#include "screen_filetest.h"
#include "button.h"
#include "tft.h"
#include "filesystem.h"
#include <stdlib.h>

```

Include dependency graph for `screen_filetest.c`:



Functions

- static void `b_back_cb` (void *button)
- static void `image_test` ()
- static void `enter` (void *screen)
- static void `leave` (void *screen)
- static void `update` (void *screen)
- `SCREEN_STRUCT` * `get_screen_filetest` ()

Variables

- static `BUTTON_STRUCT` `b_back`
- static `SCREEN_STRUCT` `screen`

7.13.1 Function Documentation

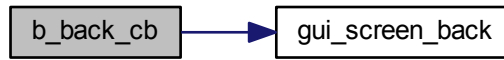
7.13.1.1 static void `b_back_cb` (void * *button*) [static]

Definition at line 27 of file `screen_filetest.c`.

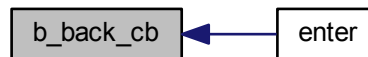
```

00028 {
00029     gui_screen_back();
00030 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



7.13.1.2 static void enter (void * screen) [static]

Definition at line 35 of file [screen_filetest.c](#).

```

00036 {
00037     tft_clear (HEX (0xABEBCD));
00038
00039     //Back button
00040     b_back.base.x1 = 10; //Start X of Button
00041     b_back.base.y1 = 200; //Start Y of Button
00042     b_back.base.x2 = AUTO; //b_back.base.x1+160; //Auto Calculate X2 with String Width
00043     b_back.base.y2 = AUTO; //Auto Calculate Y2 with String Height
00044     b_back.txtcolor = WHITE; //Set foreground color
00045     b_back.bgcolor = HEX(0xAE1010); //Set background color (Don't take 255 or 0 on at least
one channel, to make shadows possible)
00046     b_back.font = 0; //Select Font
00047     b_back.text = "Back"; //Set Text (For formatted strings take sprintf)
00048     b_back.callback = b_back_cb; //Call b_back_cb as Callback
00049     gui_button_add(&b_back); //Register Button (and run the callback from now on)
00050
00051     tft_draw_line(10, 30, 310, 30, BLACK);
00052     tft_print_line(10, 18, BLUE, TRANSPARENT, 0, "Name          D H RO Date
Time          Size");
00053
00054     int y = 33;
00055
00056     DIRECTORY_STRUCT* dir = filesystem_dir_open(".");
00057
00058     if (dir == NULL) {
00059         return;
00060     }
00061
00062     for (int i = 0; i < dir->num_files; i++) {
00063         FILE_STRUCT* file = &(dir->files[i]);
00064         tft_print_formatted(10, y,
00065             (file->fattrib & F_DIR) ? GREEN :
RED,
00066             TRANSPARENT, 0, "%-13s%c %c %s %02u%02u%02u %02u:%02u:%02u %u",
00067             file->fname,
00068             (file->fattrib & F_DIR) ? 'D' : ' ',
00069             (file->fattrib & F_HID) ? 'H' : ' ',
00070             (file->fattrib & F_RDO) ? "R " : "RW",
00071             file->fdate.day,
00072             file->fdate.month,
00073             (file->fdate.year + 1980) % 100,

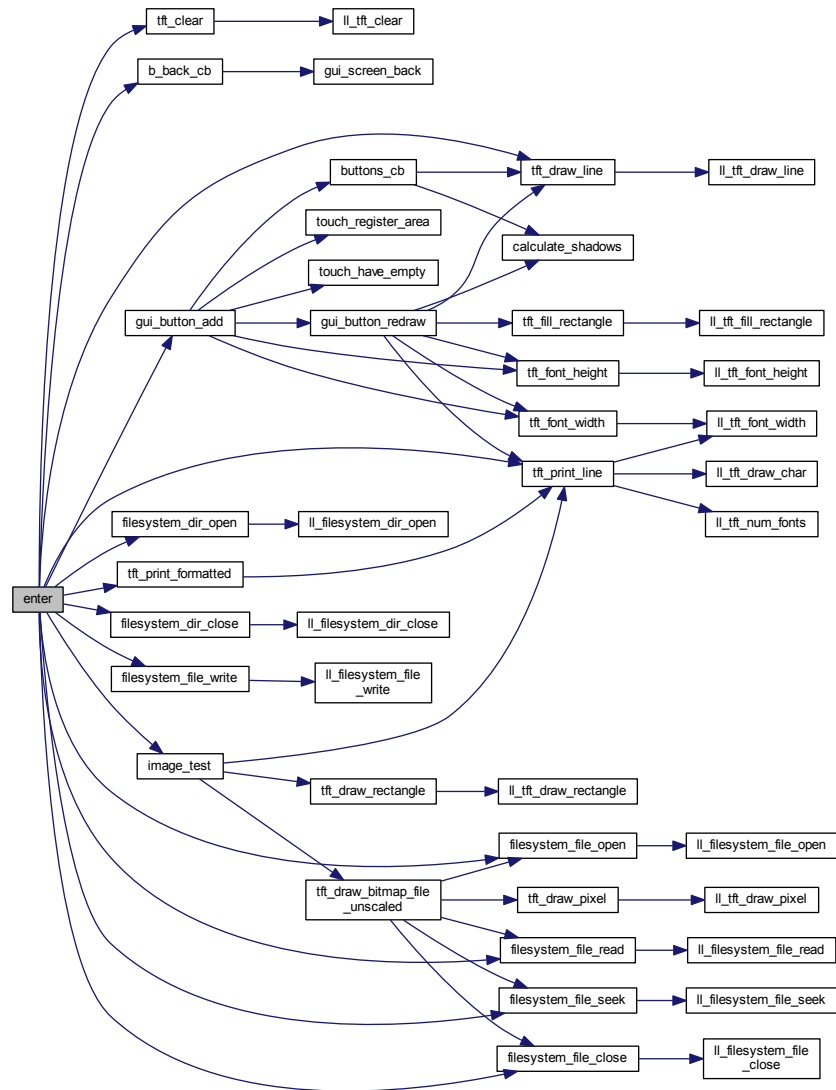
```

```

00074             file->ftime.hour,
00075             file->ftime.min,
00076             file->ftime.sec * 2,
00077             file->fsize);
00078         y += 14;
00079     }
00080     filesystem_dir_close(dir);
00081     filesystem_dir_close(dir);
00082     y += 14;
00083     FILE_HANDLE* file = filesystem_file_open("test.txt");
00084     if (file == NULL) {
00085         tft_print_line(10, y, BLUE, TRANSPARENT, 0, "Could not open test.txt")
00086     };
00087     } else {
00088         char buf [30];
00089         int size = (file->fsize > 30) ? 29 : file->fsize - 1;
00090         FILE_STATUS st = filesystem_file_read(file, buf, size);
00091         if (st == F_OK) {
00092             buf[file->fpos] = '\0';
00093             tft_print_formatted(10, y, BLUE, TRANSPARENT, 0, "test.txt
00094 contains \"%s\"", buf);
00095             long num = strtol(&(buf[file->fpos - 4]), NULL, 0);
00096             num++;
00097             y += 14;
00098             if (filesystem_file_seek(file, file->fpos - 4) !=
00099 F_OK) {
00100                 tft_print_formatted(10, y, BLUE,
00101 TRANSPARENT, 0, "Could not seek to %d", file->fpos - 4);
00102             } else {
00103                 sprintf(buf, "%04d", num);
00104                 if (filesystem_file_write(file, buf, 4) !=
00105 F_OK) {
00106                     tft_print_formatted(10, y, BLUE,
00107 TRANSPARENT, 0, "Could not write new number %d", num);
00108                 } else {
00109                     tft_print_formatted(10, y, BLUE,
00110 TRANSPARENT, 0, "New number written %d", num);
00111                 }
00112             } else {
00113                 tft_print_line(10, y, BLUE, TRANSPARENT, 0, "Could not read from
00114 test.txt");
00115             }
00116         }
00117     }
00118     filesystem_file_close(file);
00119     image_test();
00120 }
00121 }

```


Here is the call graph for this function:



7.13.1.3 static void image_test () [static]

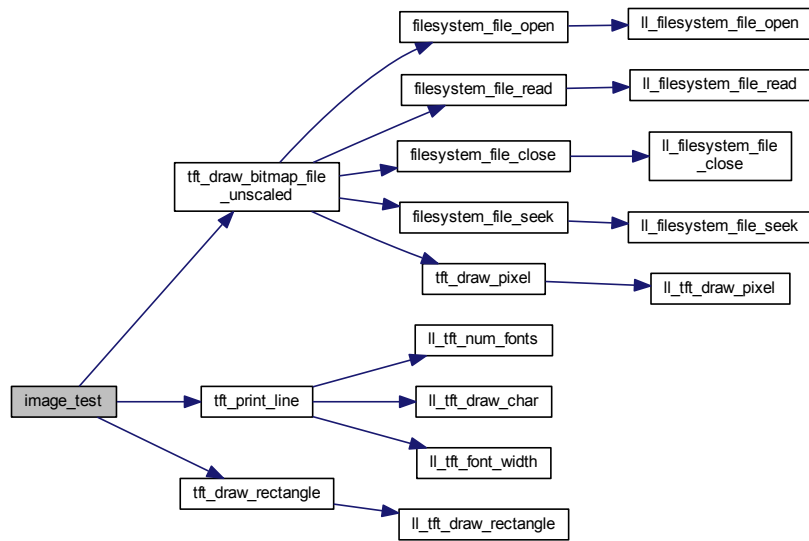
Definition at line 146 of file [screen_filetest.c](#).

```

00147 {
00148
00149
00150     if (!tft_draw_bitmap_file_unscaled(250, 170, "cpu.bmp")) {
00151         tft_print_line(10, 180, BLUE, TRANSPARENT, 0, "Could not open cpu.bmp")
00152     };
00153
00154     tft_draw_rectangle(250 - 1, 170 - 1, 250 - 1 + 64, 170 - 1 + 64,
00155         BLACK);
00156 }

```

Here is the call graph for this function:



Here is the caller graph for this function:



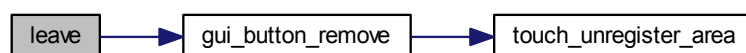
7.13.1.4 static void leave (void * screen) [static]

Definition at line 124 of file [screen_filetest.c](#).

```

00125 {
00126     gui_button_remove (&b_back);
00127 }
  
```

Here is the call graph for this function:



7.13.1.5 static void update (void * screen) [static]

Definition at line 129 of file [screen_filetest.c](#).

```
00130 {
00131 }
```

7.13.2 Variable Documentation

7.13.2.1 BUTTON_STRUCT b_back [static]

Definition at line 25 of file [screen_filetest.c](#).

7.13.2.2 SCREEN_STRUCT screen [static]

Initial value:

```
= {
    enter,
    leave,
    update
}
```

Definition at line 134 of file [screen_filetest.c](#).

7.14 screen_filetest.c

```
00001 /*****
00002  * Project:      discoverpixy
00003  * Website:     https://github.com/t-moe/discoverpixy
00004  * Authors:     Aaron Schmocker, Timo Lang
00005  * Institution: BFH Bern University of Applied Sciences
00006  * File:        common/app/screen_filetest.c
00007  *
00008  * Version History:
00009  * Date      Autor Email      SHA      Changes
00010  * 2015-05-10 timolang@gmail.com e2bce8f Added filesystem module, tests and implementation for it in
00011  * 2015-05-10 timolang@gmail.com 790f602 Added bitmap decoding/drawing example
00012  * 2015-05-10 timolang@gmail.com 21edc56 Added doxyfile (doxygen) for the common folder. Started with
00013  * 2015-05-15 timolang@gmail.com 9a16865 Added doxygen comments to filesyste, checkbox, numupdown and
00014  * 2015-05-15 timolang@gmail.com 85f1aee Changed filetest to use new bitmap draw method.
00015  *
00016  *****/
00017
00018 #include "screen_filetest.h"
00019 #include "button.h"
00020 #include "tft.h"
00021 #include "filesystem.h"
00022 #include <stdlib.h>
00023
00024
00025 static BUTTON_STRUCT b_back;
00026
00027 static void b_back_cb(void* button)
00028 {
00029     gui_screen_back();
00030 }
00031
00032
00033 static void image_test();
00034
00035 static void enter(void* screen)
00036 {
00037     tft_clear(HEX(0xBABECD));
00038
00039     //Back button
```

```

00040     b_back.base.x1 = 10; //Start X of Button
00041     b_back.base.y1 = 200; //Start Y of Button
00042     b_back.base.x2 = AUTO; //b_back.base.x1+160; //Auto Calculate X2 with String Width
00043     b_back.base.y2 = AUTO; //Auto Calculate Y2 with String Height
00044     b_back.txtcolor = WHITE; //Set foreground color
00045     b_back.bgcolor = HEX(0xAE1010); //Set background color (Don't take 255 or 0 on at least one
channel, to make shadows possible)
00046     b_back.font = 0; //Select Font
00047     b_back.text = "Back"; //Set Text (For formatted strings take sprintf)
00048     b_back.callback = b_back_cb; //Call b_back_cb as Callback
00049     gui_button_add(&b_back); //Register Button (and run the callback from now on)
00050
00051     tft_draw_line(10, 30, 310, 30, BLACK);
00052     tft_print_line(10, 18, BLUE, TRANSPARENT, 0, "Name          D H RO Date
Time          Size");
00053
00054     int y = 33;
00055
00056     DIRECTORY_STRUCT* dir = filesystem_dir_open(".");
00057
00058     if (dir == NULL) {
00059         return;
00060     }
00061
00062     for (int i = 0; i < dir->num_files; i++) {
00063         FILE_STRUCT* file = &(dir->files[i]);
00064         tft_print_formatted(10, y,
00065             (file->fattrib & F_DIR) ? GREEN :
RED,
00066             TRANSPARENT, 0, "%-13s%c %c %s %02u%02u%02u %02u:%02u:%02u %u",
00067             file->fname,
00068             (file->fattrib & F_DIR) ? 'D' : ' ',
00069             (file->fattrib & F_HID) ? 'H' : ' ',
00070             (file->fattrib & F_RDO) ? "R" : "RW",
00071             file->fdate.day,
00072             file->fdate.month,
00073             (file->fdate.year + 1980) % 100,
00074             file->ftime.hour,
00075             file->ftime.min,
00076             file->ftime.sec * 2,
00077             file->fsize);
00078         y += 14;
00079     }
00080
00081     filesystem_dir_close(dir);
00082
00083     y += 14;
00084
00085     FILE_HANDLE* file = filesystem_file_open("test.txt");
00086
00087     if (file == NULL) {
00088         tft_print_line(10, y, BLUE, TRANSPARENT, 0, "Could not open test.txt")
;
00089     } else {
00090         char buf [30];
00091         int size = (file->fsize > 30) ? 29 : file->fsize - 1;
00092         FILE_STATUS st = filesystem_file_read(file, buf, size);
00093
00094         if (st == F_OK) {
00095             buf[file->fpos] = '\0';
00096             tft_print_formatted(10, y, BLUE, TRANSPARENT, 0, "test.txt
contains \"%s\"", buf);
00097             long num = strtol(&(buf[file->fpos - 4]), NULL, 0);
00098             num++;
00099
00100             y += 14;
00101
00102             if (filesystem_file_seek(file, file->fpos - 4) !=
F_OK) {
00103                 tft_print_formatted(10, y, BLUE,
TRANSPARENT, 0, "Could not seek to %d", file->fpos - 4);
00104             } else {
00105                 sprintf(buf, "%04d", num);
00106
00107                 if (filesystem_file_write(file, buf, 4) !=
F_OK) {
00108                     tft_print_formatted(10, y, BLUE,
TRANSPARENT, 0, "Could not write new number %d", num);
00109                 } else {
00110                     tft_print_formatted(10, y, BLUE,
TRANSPARENT, 0, "New number written %d", num);
00111                 }
00112             }
00113         } else {
00114             tft_print_line(10, y, BLUE, TRANSPARENT, 0, "Could not read from
test.txt");
00115         }

```

```

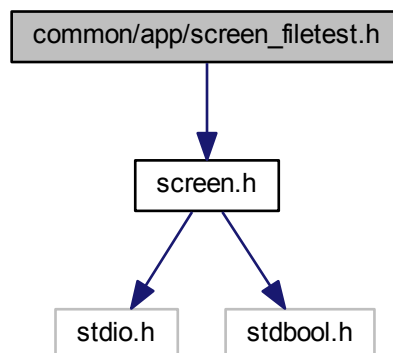
00116
00117     }
00118
00119     filesystem_file_close(file);
00120
00121     image_test();
00122 }
00123
00124 static void leave(void* screen)
00125 {
00126     gui_button_remove(&b_back);
00127 }
00128
00129 static void update(void* screen)
00130 {
00131 }
00132
00133
00134 static SCREEN_STRUCT screen = {
00135     enter,
00136     leave,
00137     update
00138 };
00139
00140
00141 SCREEN_STRUCT* get_screen_filetest()
00142 {
00143     return &screen;
00144 }
00145
00146 static void image_test()
00147 {
00148
00149
00150     if (!tft_draw_bitmap_file_unscaled(250, 170, "cpu.bmp")) {
00151         tft_print_line(10, 180, BLUE, TRANSPARENT, 0, "Could not open cpu.bmp"
00152     );
00153     }
00154     tft_draw_rectangle(250 - 1, 170 - 1, 250 - 1 + 64, 170 - 1 + 64,
00155     BLACK);
00156 }

```

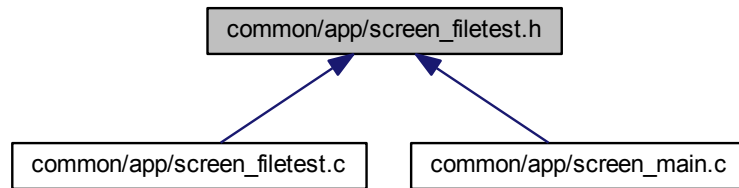
7.15 common/app/screen_filetest.h File Reference

```
#include "screen.h"
```

Include dependency graph for screen_filetest.h:



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



Functions

- `SCREEN_STRUCT * get_screen_filetest ()`

7.16 screen_filetest.h

```

00001 /*****
00002  *****/
00003 * Project:      discoverpixy
00004 * Website:     https://github.com/t-moe/discoverpixy
00005 * Authors:     Aaron Schmocker, Timo Lang
00006 * Institution: BFH Bern University of Applied Sciences
00007 * File:        common/app/screen_filetest.h
00008 *
00009 * Version History:
00010 * Date      Autor Email      SHA      Changes
00011 * 2015-05-10 timolang@gmail.com e2bce8f Added filesystem module, tests and implementation for it in
00012 *            emulator.
00013 * 2015-05-10 timolang@gmail.com 21edc56 Added doxyfile (doxygen) for the common folder. Started with
00014 *            doxygen comments for app and tft module.
00015 * 2015-05-15 timolang@gmail.com 9a16865 Added doxygen comments to filesyste, checkbox, numupdown and
00016 *            screen module. And some minor changes to the other modules.
00017 *
00018 *****/
00019
00020 #include "screen.h"
00021
00022
00023
00024
00025 SCREEN_STRUCT* get_screen_filetest();
00026

```

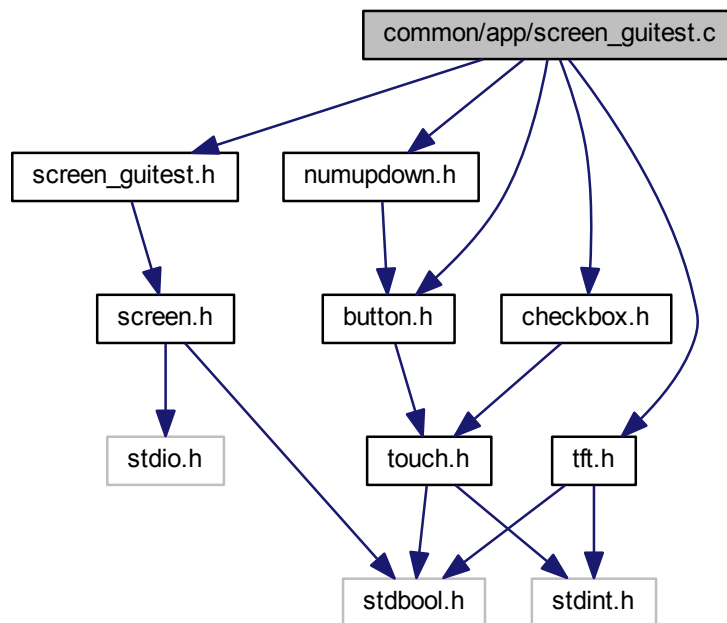
7.17 common/app/screen_guitest.c File Reference

```

#include "screen_guitest.h"
#include "button.h"
#include "tft.h"
#include "checkbox.h"
#include "numupdown.h"

```

Include dependency graph for screen_guitest.c:



Functions

- static void `checkboxCB` (void *checkbox, bool checked)
- static void `b_back_cb` (void *button)
- static void `n_updown_cb` (void *numupdown, int16_t value)
- static void `touchCB` (void *touchArea, `TOUCH_ACTION` triggeredAction)
- static void `enter` (void *screen)
- static void `leave` (void *screen)
- static void `update` (void *screen)
- `SCREEN_STRUCT` * `get_screen_guitest` ()

Variables

- static `BUTTON_STRUCT` `b_back`
- static `TOUCH_AREA_STRUCT` `a_area`
- static `CHECKBOX_STRUCT` `c_cbox`
- static `NUMUPDOWN_STRUCT` `n_updown`
- static `SCREEN_STRUCT` `screen`

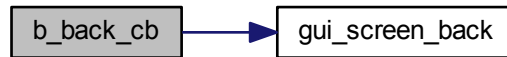
7.17.1 Function Documentation

7.17.1.1 static void `b_back_cb` (void * *button*) [static]

Definition at line 35 of file `screen_guitest.c`.

```
00036 {  
00037     gui_screen_back();  
00038 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



7.17.1.2 static void checkboxCB (void * *checkbox*, bool *checked*) [static]

Definition at line 30 of file [screen_guitest.c](#).

```
00031 {  
00032     printf("Checkbox %s\n", (checked ? "checked" : "unchecked"));  
00033 }
```

Here is the caller graph for this function:



7.17.1.3 static void enter (void * *screen*) [static]

Definition at line 75 of file [screen_guitest.c](#).

```
00076 {  
00077     tft_clear (HEX (0xA6FD9A) );  
00078 }
```

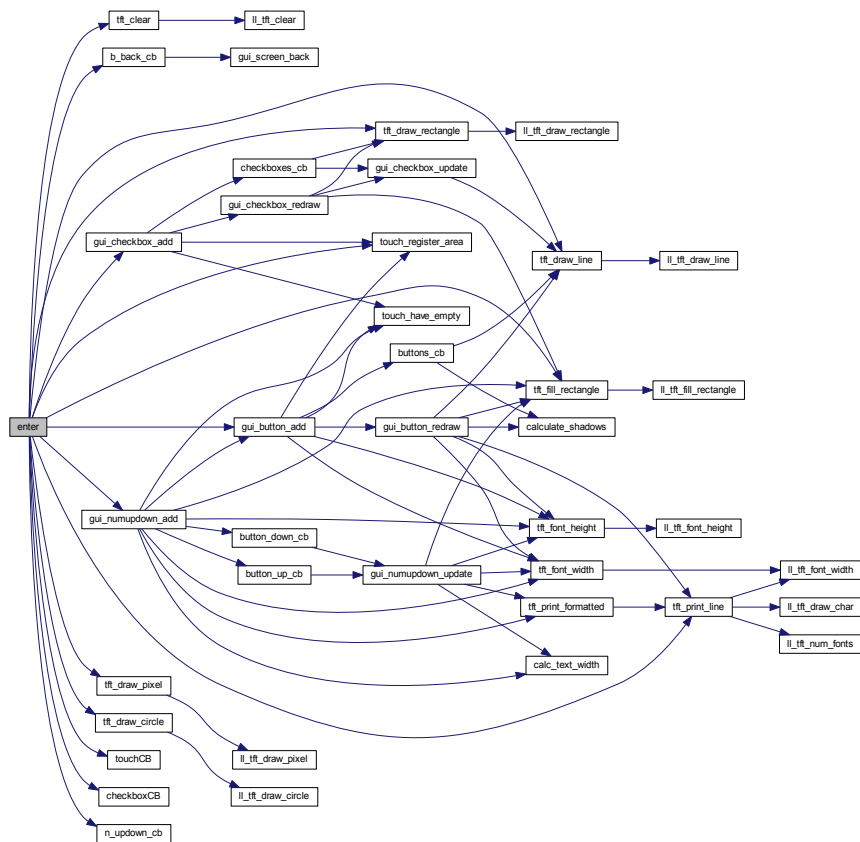


```

00079     //Back button
00080     b_back.base.x1 = 10; //Start X of Button
00081     b_back.base.y1 = 10; //Start Y of Button
00082     b_back.base.x2 = AUTO; //b_back.base.x1+160; //Auto Calculate X2 with String Width
00083     b_back.base.y2 = AUTO; //Auto Calculate Y2 with String Height
00084     b_back.txtcolor = WHITE; //Set foreground color
00085     b_back.bgcolor = HEX(0xAE1010); //Set background color (Don't take 255 or 0 on at least
one channel, to make shadows possible)
00086     b_back.font = 0; //Select Font
00087     b_back.text = "Back"; //Set Text (For formatted strings take sprintf)
00088     b_back.callback = b_back_cb; //Call b_back_cb as Callback
00089     gui_button_add(&b_back); //Register Button (and run the callback from now on)
00090
00091
00092     //tft test
00093     tft_draw_pixel(0, 0, BLACK);
00094     tft_draw_pixel(319, 239, BLACK);
00095     tft_draw_pixel(10, 210, BLUE);
00096     tft_draw_pixel(12, 210, BLUE);
00097     tft_draw_rectangle(40, 100, 60, 235, BLUE);
00098     tft_fill_rectangle(100, 215, 200, 225, GREEN);
00099     tft_draw_line(10, 50, 310, 225, RGB(0xFF, 0, 0xFF));
00100     tft_draw_circle(10, 10, 100, RED);
00101     tft_print_line(30, 130, RED, BLUE, 0, "Hallo");
00102
00103
00104
00105     //Area test
00106     a_area.hookedActions = PEN_DOWN | PEN_UP |
PEN_MOVE | PEN_ENTER | PEN_LEAVE;
00107     a_area.x1 = 130;
00108     a_area.y1 = 30;
00109     a_area.x2 = 200;
00110     a_area.y2 = 60;
00111     a_area.callback = touchCB;
00112     touch_register_area(&a_area);
00113
00114
00115
00116     //Checkbox test
00117     c_cbox.base.x1 = 220;
00118     c_cbox.base.y1 = 45;
00119     c_cbox.base.x2 = c_cbox.base.x1 + 16;
00120     c_cbox.base.y2 = c_cbox.base.y1 + 16;
00121     c_cbox.fgcolor = GREEN;
00122     c_cbox.checked = true;
00123     c_cbox.callback = checkboxCB;
00124     gui_checkbox_add(&c_cbox);
00125
00126
00127     //Num up down test
00128     n_updown.x = 200;
00129     n_updown.y = 120;
00130     n_updown.fgcolor = RED;
00131     n_updown.value = -3;
00132     n_updown.max = 11;
00133     n_updown.min = -5;
00134     n_updown.callback = n_updown_cb;
00135     gui_numupdown_add(&n_updown);
00136
00137 }

```

Here is the call graph for this function:

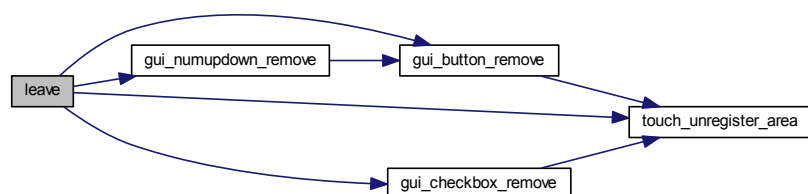


7.17.1.4 static void leave (void * screen) [static]

Definition at line 139 of file [screen_guitest.c](#).

```
00140 {
00141     gui_button_remove (&b_back);
00142     gui_checkbox_remove (&c_cbox);
00143     gui_numupdown_remove (&n_updown);
00144     touch_unregister_area (&a_area);
00145 }
```

Here is the call graph for this function:

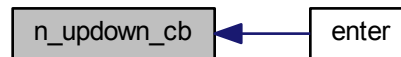


7.17.1.5 static void n_updown_cb (void * numupdown, int16_t value) [static]

Definition at line 40 of file [screen_guitest.c](#).

```
00041 {  
00042     printf("New NumUpDown Value %d\n", value);  
00043 }
```

Here is the caller graph for this function:

**7.17.1.6 static void touchCB (void * touchArea, TOUCH_ACTION triggeredAction) [static]**

Definition at line 45 of file [screen_guitest.c](#).

```
00046 {  
00047     switch (triggeredAction) {  
00048     case PEN_DOWN:  
00049         printf("action PEN_DOWN\n");  
00050         break;  
00051     case PEN_UP:  
00052         printf("action PEN_UP\n");  
00053         break;  
00054     case PEN_MOVE:  
00055         printf("action PEN_MOVE\n");  
00056         break;  
00057     case PEN_ENTER:  
00058         printf("action PEN_ENTER\n");  
00059         break;  
00060     case PEN_LEAVE:  
00061         printf("action PEN_LEAVE\n");  
00062         break;  
00063     default:  
00064         printf("action %s\n", triggeredAction);  
00065         break;  
00066     }  
00067 }
```

Here is the caller graph for this function:



7.17.1.7 static void update (void * screen) [static]

Definition at line 147 of file [screen_guitest.c](#).

```
00148 {
00149     //gui_button_redraw(&b_back); //only needed if button is overdrawn by others
00150     //.... for the other elements as well
00151 }
```

7.17.2 Variable Documentation

7.17.2.1 TOUCH_AREA_STRUCT a_area [static]

Definition at line 26 of file [screen_guitest.c](#).

7.17.2.2 BUTTON_STRUCT b_back [static]

Definition at line 25 of file [screen_guitest.c](#).

7.17.2.3 CHECKBOX_STRUCT c_cbox [static]

Definition at line 27 of file [screen_guitest.c](#).

7.17.2.4 NUMUPDOWN_STRUCT n_updown [static]

Definition at line 28 of file [screen_guitest.c](#).

7.17.2.5 SCREEN_STRUCT screen [static]

Initial value:

```
= {
    enter,
    leave,
    update
}
```

Definition at line 154 of file [screen_guitest.c](#).

7.18 screen_guitest.c

```
00001 /*****
00002 * Project:      discoverpixy
00003 * Website:      https://github.com/t-moe/discoverpixy
00004 * Authors:      Aaron Schmocker, Timo Lang
00005 * Institution:  BFH Bern University of Applied Sciences
00006 * File:         common/app/screen_guitest.c
00007 *
00008 * Version History:
00009 * Date          Autor Email          SHA          Changes
00010 * 2015-04-27    timolang@gmail.com    cf72baa    Introduced a Screen (sub) module and divided app into multiple
00011 *              timolang@gmail.com    76ea9d8    Added num up down support.
00012 * 2015-05-09    timolang@gmail.com    c652b6b    Improved Emulator Gui
00013 * 2015-05-29    aaron@duckpond.ch     7d2d1a1    Implemented basic sampling and averaging of touch coordinates
00014 *              timolang@gmail.com    eb573bc    Finalized calibration. Fixed a bug in screen module.
00015 * 2015-06-02    timolang@gmail.com    da34bce    Fixed all printf related problems on discovery using
00016 *              workarounds and newlib nano-instead of newlib
00017 *****/
```

```

00018 *****/
00019 #include "screen_guittest.h"
00020 #include "button.h"
00021 #include "tft.h"
00022 #include "checkboxbox.h"
00023 #include "numupdown.h"
00024
00025 static BUTTON_STRUCT b_back;
00026 static TOUCH_AREA_STRUCT a_area;
00027 static CHECKBOX_STRUCT c_cbox;
00028 static NUMUPDOWN_STRUCT n_updown;
00029
00030 static void checkboxCB(void* checkbox, bool checked)
00031 {
00032     printf("Checkbox %s\n", (checked ? "checked" : "unchecked"));
00033 }
00034
00035 static void b_back_cb(void* button)
00036 {
00037     gui_screen_back();
00038 }
00039
00040 static void n_updown_cb(void* numupdown, int16_t value)
00041 {
00042     printf("New NumUpDown Value %d\n", value);
00043 }
00044
00045 static void touchCB(void* touchArea, TOUCH_ACTION triggeredAction)
00046 {
00047
00048     switch (triggeredAction) {
00049     case PEN_DOWN:
00050         printf("action PEN_DOWN\n");
00051         break;
00052
00053     case PEN_UP:
00054         printf("action PEN_UP\n");
00055         break;
00056
00057     case PEN_MOVE:
00058         printf("action PEN_MOVE\n");
00059         break;
00060
00061     case PEN_ENTER:
00062         printf("action PEN_ENTER\n");
00063         break;
00064
00065     case PEN_LEAVE:
00066         printf("action PEN_LEAVE\n");
00067         break;
00068
00069     default:
00070         printf("action %s\n", triggeredAction);
00071         break;
00072     }
00073 }
00074
00075 static void enter(void* screen)
00076 {
00077     tft_clear(HEX(0xA6FD9A));
00078
00079     //Back button
00080     b_back.base.x1 = 10; //Start X of Button
00081     b_back.base.y1 = 10; //Start Y of Button
00082     b_back.base.x2 = AUTO; //b_back.base.x1+160; //Auto Calculate X2 with String Width
00083     b_back.base.y2 = AUTO; //Auto Calculate Y2 with String Height
00084     b_back.txtcolor = WHITE; //Set foreground color
00085     b_back.bgcolor = HEX(0xAE1010); //Set background color (Don't take 255 or 0 on at least one
channel, to make shadows possible)
00086     b_back.font = 0; //Select Font
00087     b_back.text = "Back"; //Set Text (For formatted strings take sprintf)
00088     b_back.callback = b_back_cb; //Call b_back_cb as Callback
00089     gui_button_add(&b_back); //Register Button (and run the callback from now on)
00090
00091
00092     //tft test
00093     tft_draw_pixel(0, 0, BLACK);
00094     tft_draw_pixel(319, 239, BLACK);
00095     tft_draw_pixel(10, 210, BLUE);
00096     tft_draw_pixel(12, 210, BLUE);
00097     tft_draw_rectangle(40, 100, 60, 235, BLUE);
00098     tft_fill_rectangle(100, 215, 200, 225, GREEN);
00099     tft_draw_line(10, 50, 310, 225, RGB(0xFF, 0, 0xFF));
00100     tft_draw_circle(10, 10, 100, RED);
00101     tft_print_line(30, 130, RED, BLUE, 0, "Hallo");
00102

```

```

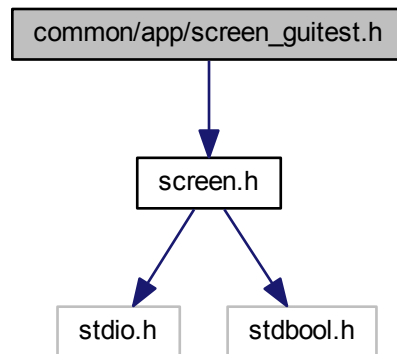
00103
00104
00105     //Area test
00106     a_area.hookedActions = PEN_DOWN | PEN_UP |
PEN_MOVE | PEN_ENTER | PEN_LEAVE;
00107     a_area.x1 = 130;
00108     a_area.y1 = 30;
00109     a_area.x2 = 200;
00110     a_area.y2 = 60;
00111     a_area.callback = touchCB;
00112     touch_register_area(&a_area);
00113
00114
00115
00116     //Checkbox test
00117     c_cbox.base.x1 = 220;
00118     c_cbox.base.y1 = 45;
00119     c_cbox.base.x2 = c_cbox.base.x1 + 16;
00120     c_cbox.base.y2 = c_cbox.base.y1 + 16;
00121     c_cbox.fgcolor = GREEN;
00122     c_cbox.checked = true;
00123     c_cbox.callback = checkboxCB;
00124     gui_checkbox_add(&c_cbox);
00125
00126
00127     //Num up down test
00128     n_updown.x = 200;
00129     n_updown.y = 120;
00130     n_updown.fgcolor = RED;
00131     n_updown.value = -3;
00132     n_updown.max = 11;
00133     n_updown.min = -5;
00134     n_updown.callback = n_updown_cb;
00135     gui_numupdown_add(&n_updown);
00136
00137 }
00138
00139 static void leave(void* screen)
00140 {
00141     gui_button_remove(&b_back);
00142     gui_checkbox_remove(&c_cbox);
00143     gui_numupdown_remove(&n_updown);
00144     touch_unregister_area(&a_area);
00145 }
00146
00147 static void update(void* screen)
00148 {
00149     //gui_button_redraw(&b_back); //only needed if button is overdrawn by others
00150     //.... for the other elements as well
00151 }
00152
00153
00154 static SCREEN_STRUCT screen = {
00155     enter,
00156     leave,
00157     update
00158 };
00159
00160
00161 SCREEN_STRUCT* get_screen_guitest()
00162 {
00163     return &screen;
00164 }

```

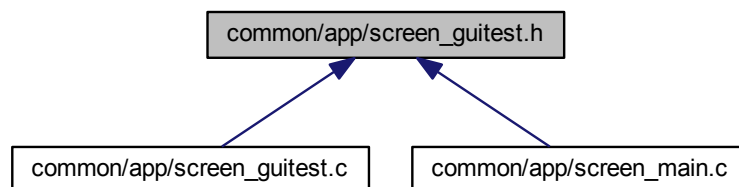
7.19 common/app/screen_guitest.h File Reference

```
#include "screen.h"
```

Include dependency graph for screen_guitest.h:



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



Functions

- `SCREEN_STRUCT * get_screen_guitest ()`

7.20 screen_guitest.h

```

00001 /*****
00002  * Project:      discoverpixy
00003  * Website:     https://github.com/t-moe/discoverpixy
00004  * Authors:     Aaron Schmocker, Timo Lang
00005  * Institution:  BFH Bern University of Applied Sciences
00006  * File:        common/app/screen_guitest.h
00007  *
00008  * Version History:
00009  * Date          Autor Email          SHA      Changes
00010  * 2015-04-27    timolang@gmail.com    cf72baa  Introduced a Screen (sub) module and divided app into multiple
00011  * 2015-05-10    timolang@gmail.com    21edc56  Added doxyfile (doxygen) for the common folder. Started with
  
```

```

doxygen comments for app and tft module.
00012 * 2015-05-15 timolang@gmail.com 9a16865 Added doxygen comments to filesystem, checkbox, numupdown and
screen module. And some minor changes to the other modules.
00013 *
00014 *****/
00015
00016 #include "screen.h"
00017
00018
00023
00029
00030
00036 SCREEN_STRUCT* get_screen_guitest ();
00037

```

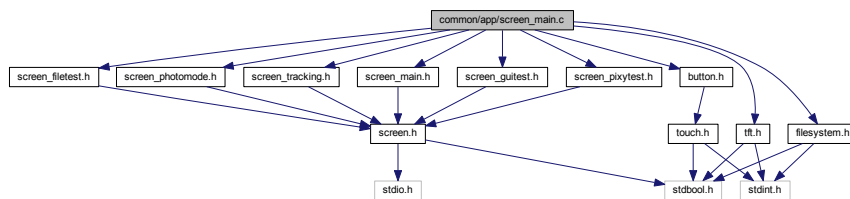
7.21 common/app/screen_main.c File Reference

```

#include "screen_main.h"
#include "screen_guitest.h"
#include "screen_pixytest.h"
#include "screen_filetest.h"
#include "screen_photomode.h"
#include "screen_tracking.h"
#include "button.h"
#include "tft.h"
#include "filesystem.h"

```

Include dependency graph for screen_main.c:



Macros

- #define [X_TAB](#) 97
- #define [BUTTON_SPACING](#) 7
- #define [Y_FIRST](#) 60
- #define [Y_SECOND](#) [Y_FIRST](#)+25
- #define [Y_THIRD](#) [Y_SECOND](#)+25

Functions

- static void [b_our_tracking_cb](#) (void *button)
- static void [b_ref_tracking_cb](#) (void *button)
- static void [b_photo_mode_cb](#) (void *button)
- static void [b_guitest_cb](#) (void *button)
- static void [b_filetest_cb](#) (void *button)
- static void [b_pixytest_cb](#) (void *button)
- static void [enter](#) (void *screen)
- static void [leave](#) (void *screen)
- static void [update](#) (void *screen)
- [SCREEN_STRUCT](#) * [get_screen_main](#) ()

Variables

- [BUTTON_STRUCT b_guitest](#)
- [BUTTON_STRUCT b_pixytest](#)
- [BUTTON_STRUCT b_filetest](#)
- [BUTTON_STRUCT b_our_tracking](#)
- [BUTTON_STRUCT b_ref_tracking](#)
- [BUTTON_STRUCT b_photo_mode](#)
- static [SCREEN_STRUCT screen](#)

7.21.1 Macro Definition Documentation

7.21.1.1 `#define BUTTON_SPACING 7`

7.21.1.2 `#define X_TAB 97`

7.21.1.3 `#define Y_FIRST 60`

7.21.1.4 `#define Y_SECOND Y_FIRST+25`

7.21.1.5 `#define Y_THIRD Y_SECOND+25`

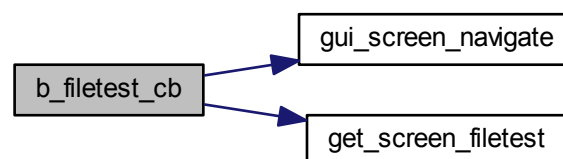
7.21.2 Function Documentation

7.21.2.1 `static void b_filetest_cb ( void * button )` `[static]`

Definition at line 60 of file [screen_main.c](#).

```
00061 {  
00062     gui\_screen\_navigate(get\_screen\_filetest());  
00063 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



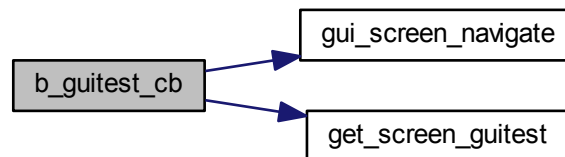
7.21.2.2 static void b_guitest_cb (void * *button*) [static]

Definition at line 55 of file [screen_main.c](#).

```

00056 {
00057     gui_screen_navigate (get_screen_guitest () );
00058 }
  
```

Here is the call graph for this function:



Here is the caller graph for this function:



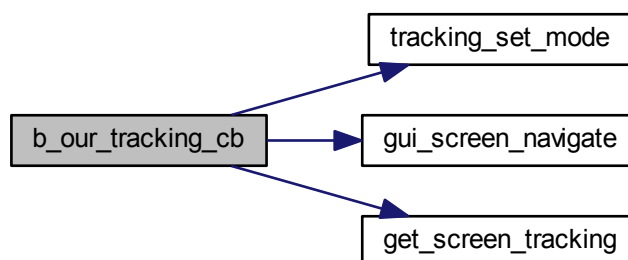
7.21.2.3 static void b_our_tracking_cb (void * *button*) [static]

Definition at line 38 of file [screen_main.c](#).

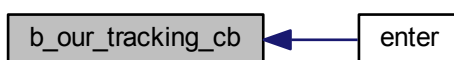
```

00039 {
00040     tracking_set_mode (OUR_TRACKING);
00041     gui_screen_navigate (get_screen_tracking());
00042 }
  
```

Here is the call graph for this function:



Here is the caller graph for this function:

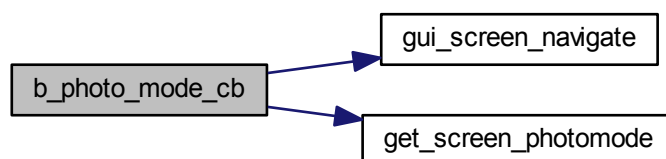


7.21.2.4 static void b_photo_mode_cb(void * *button*) [static]

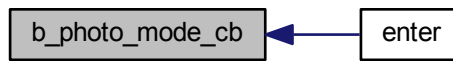
Definition at line 50 of file [screen_main.c](#).

```
00051 {  
00052     gui_screen_navigate( get_screen_photomode() );  
00053 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



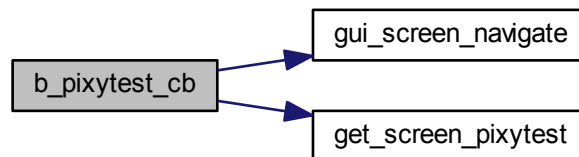
7.21.2.5 static void b_pixytest_cb (void * *button*) [static]

Definition at line 65 of file [screen_main.c](#).

```

00066 {
00067     gui_screen_navigate (get_screen_pixytest ());
00068 }
  
```

Here is the call graph for this function:



Here is the caller graph for this function:



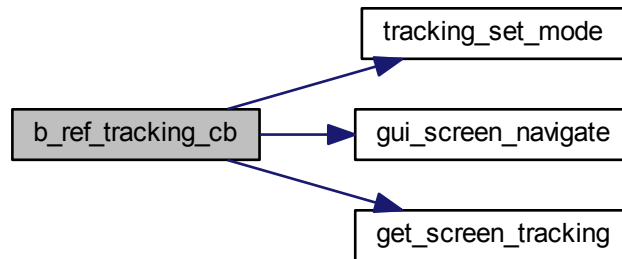
7.21.2.6 static void b_ref_tracking_cb (void * *button*) [static]

Definition at line 44 of file [screen_main.c](#).

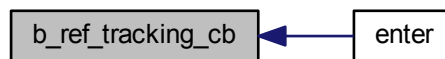
```

00045 {
00046     tracking_set_mode (REFERENCE_TRACKING);
00047     gui_screen_navigate (get_screen_tracking ());
00048 }
  
```

Here is the call graph for this function:



Here is the caller graph for this function:



7.21.2.7 static void enter (void * screen) [static]

Definition at line 71 of file [screen_main.c](#).

```

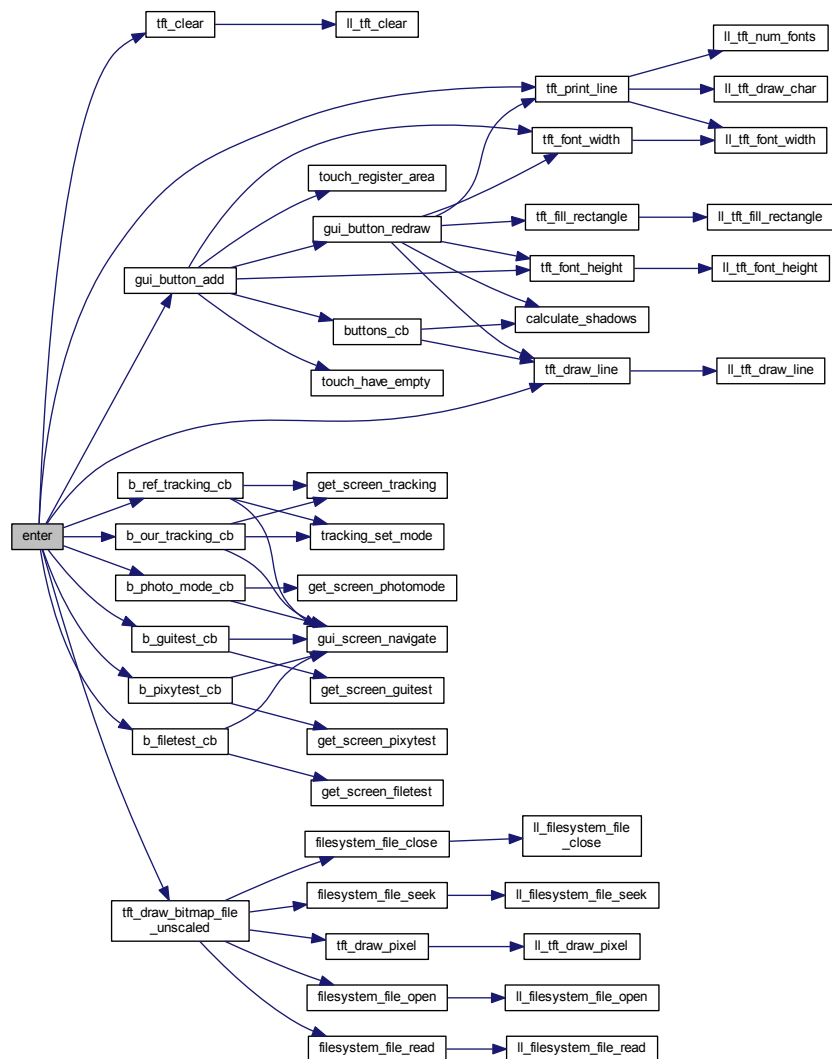
00072 {
00073     tft_clear(WHITE);
00074
00075     //Heading
00076     tft_print_line(10, 10, BLUE, TRANSPARENT, 1, "Discoverpixy");
00077     tft_draw_line(0, 40, 319, 40, BLACK);
00078
00079     #define X_TAB          97
00080     #define BUTTON_SPACING 7
00081
00082     //First line of buttons
00083     #define Y_FIRST        60
00084     tft_print_line(10, Y_FIRST, BLACK, TRANSPARENT, 0, "Tracking:");
00085
00086     b_our_tracking.base.x1 = X_TAB; //Start X of Button
00087     b_our_tracking.base.y1 = Y_FIRST - 3; //Start Y of Button
00088     b_our_tracking.base.x2 = AUTO; //Auto Calculate X2 with String Width
00089     b_our_tracking.base.y2 = AUTO; //Auto Calculate Y2 with String Height
00090     b_our_tracking.txtcolor = WHITE; //Set foreground color
00091     b_our_tracking.bgcolor = HEX(0xE30535); //Set background color (Don't take 255
or 0 on at least one channel, to make shadows possible)
00092     b_our_tracking.font = 0; //Select Font
00093     b_our_tracking.text = "Our Tracking"; //Set Text (For formatted strings take sprintf)
00094     b_our_tracking.callback = b_our_tracking_cb; //Call
b_our_tracking when the button get's pressed
00095     gui_button_add(&b_our_tracking); //Register Button (and run the callback
from now on)
00096
00097
00098     b_ref_tracking.base.x1 = b_our_tracking.
base.x2 + BUTTON_SPACING;
  
```

```

00099     b_ref_tracking.base.y1 = Y_FIRST - 3;
00100     b_ref_tracking.base.x2 = AUTO;
00101     b_ref_tracking.base.y2 = AUTO;
00102     b_ref_tracking.txtcolor = WHITE;
00103     b_ref_tracking.bgcolor = HEX(0xFF2151);
00104     b_ref_tracking.font = 0;
00105     b_ref_tracking.text = "Ref Tracking";
00106     b_ref_tracking.callback = b_ref_tracking_cb;
00107     gui_button_add(&b_ref_tracking);
00108
00109     //Second line of buttons
00110 #define Y_SECOND Y_FIRST+25
00111     tft_print_line(10, Y_SECOND, BLACK, TRANSPARENT, 0, "Photo mode:"
00112 );
00113     b_photo_mode.base.x1 = X_TAB;
00114     b_photo_mode.base.y1 = Y_SECOND - 3;
00115     b_photo_mode.base.x2 = AUTO;
00116     b_photo_mode.base.y2 = AUTO;
00117     b_photo_mode.txtcolor = WHITE;
00118     b_photo_mode.bgcolor = HEX(0x21B1FF);
00119     b_photo_mode.font = 0;
00120     b_photo_mode.text = "Photo Mode";
00121     b_photo_mode.callback = b_photo_mode_cb;
00122     gui_button_add(&b_photo_mode);
00123
00124
00125     //Third line of buttons
00126 #define Y_THIRD Y_SECOND+25
00127     tft_print_line(10, Y_THIRD, BLACK, TRANSPARENT, 0, "Tests:");
00128
00129     b_guittest.base.x1 = X_TAB;
00130     b_guittest.base.y1 = Y_THIRD - 3;
00131     b_guittest.base.x2 = AUTO;
00132     b_guittest.base.y2 = AUTO;
00133     b_guittest.txtcolor = BLACK;
00134     b_guittest.bgcolor = HEX(0x00FA21);
00135     b_guittest.font = 0;
00136     b_guittest.text = "Gui & Tft";
00137     b_guittest.callback = b_guittest_cb;
00138     gui_button_add(&b_guittest);
00139
00140
00141     b_pixytest.base.x1 = b_guittest.base.x2 +
00142     BUTTON_SPACING;
00143     b_pixytest.base.y1 = Y_THIRD - 3;
00144     b_pixytest.base.x2 = AUTO;
00145     b_pixytest.base.y2 = AUTO;
00146     b_pixytest.txtcolor = BLACK;
00147     b_pixytest.bgcolor = HEX(0x00FA96);
00148     b_pixytest.font = 0;
00149     b_pixytest.text = "Pixy";
00150     b_pixytest.callback = b_pixytest_cb;
00151     gui_button_add(&b_pixytest);
00152
00153     b_filetest.base.x1 = b_pixytest.base.x2 +
00154     BUTTON_SPACING;
00155     b_filetest.base.y1 = Y_THIRD - 3;
00156     b_filetest.base.x2 = AUTO;
00157     b_filetest.base.y2 = AUTO;
00158     b_filetest.txtcolor = BLACK;
00159     b_filetest.bgcolor = HEX(0x00FAC4);
00160     b_filetest.font = 0;
00161     b_filetest.text = "File";
00162     b_filetest.callback = b_filetest_cb;
00163     gui_button_add(&b_filetest);
00164
00165     //Bottom line
00166     tft_draw_line(0, 145, 319, 145, BLACK);
00167     tft_print_line(10, 150, BLUE, TRANSPARENT, 0, "Powered by");
00168     tft_draw_bitmap_file_unscaled(10, 165, "pixy_small.bmp");
00169     tft_draw_bitmap_file_unscaled(165, 165, "stm_small.bmp");
00170
00171 }

```

Here is the call graph for this function:



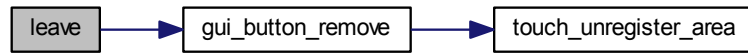
7.21.2.8 static void leave (void * *screen*) [static]

Definition at line 173 of file [screen_main.c](#).

```

00174 {
00175     gui_button_remove (&b_our_tracking);
00176     gui_button_remove (&b_ref_tracking);
00177     gui_button_remove (&b_photo_mode);
00178     gui_button_remove (&b_guitest);
00179     gui_button_remove (&b_pixytest);
00180     gui_button_remove (&b_filetest);
00181 }
  
```

Here is the call graph for this function:



7.21.2.9 static void update (void * screen) [static]

Definition at line 183 of file [screen_main.c](#).

```

00184 {
00185     //gui_button_redraw(&b_guitest); //only needed if button is overdrawn by others
00186 }
  
```

7.21.3 Variable Documentation

7.21.3.1 BUTTON_STRUCT b_filetest

Definition at line 31 of file [screen_main.c](#).

7.21.3.2 BUTTON_STRUCT b_guitest

Definition at line 29 of file [screen_main.c](#).

7.21.3.3 BUTTON_STRUCT b_our_tracking

Definition at line 33 of file [screen_main.c](#).

7.21.3.4 BUTTON_STRUCT b_photo_mode

Definition at line 35 of file [screen_main.c](#).

7.21.3.5 BUTTON_STRUCT b_pixytest

Definition at line 30 of file [screen_main.c](#).

7.21.3.6 BUTTON_STRUCT b_ref_tracking

Definition at line 34 of file [screen_main.c](#).

7.21.3.7 SCREEN_STRUCT screen [static]

Initial value:

```

= {
    enter,
    leave,
    update
}
  
```


Definition at line 189 of file screen_main.c.

7.22 screen_main.c

```

00001 /*****
00002  * Project:      discoverpixy
00003  * Website:     https://github.com/t-moe/discoverpixy
00004  * Authors:     Aaron Schmocker, Timo Lang
00005  * Institution:  BFH Bern University of Applied Sciences
00006  * File:        common/app/screen_main.c
00007  *
00008  * Version History:
00009  * Date          Autor Email          SHA          Changes
00010  * 2015-04-27    timolang@gmail.com    cf72baa    Introduced a Screen (sub) module and divided app into multiple
00011  *              screens.
00012  * 2015-05-10    timolang@gmail.com    e2bce8f    Added filesystem module, tests and implementation for it in
00013  *              emulator.
00014  * 2015-05-15    timolang@gmail.com    27c09ba    Redesigned main menu. Moved stuff from pixytest to a new helper
00015  *              file and to the new "photo mode"-screen.
00016  * 2015-05-16    timolang@gmail.com    e46314b    Added Tracking Screen and implemented "Reference Tracking" and
00017  *              "Color Region Selection"
00018  * 2015-06-01    aaron@duckpond.ch     caa7b5c    Added IRQ for user button, fixed some touchproblems.
00019  * 2015-06-01    timolang@gmail.com    3155f42    Fixed mainscreen layout.
00020  *****/
00021 #include "screen_main.h"
00022 #include "screen_guitest.h"
00023 #include "screen_pixytest.h"
00024 #include "screen_filetest.h"
00025 #include "screen_photomode.h"
00026 #include "screen_tracking.h"
00027 #include "button.h"
00028 #include "tft.h"
00029 #include "filesystem.h"
00030
00031 BUTTON_STRUCT b_guitest;
00032 BUTTON_STRUCT b_pixytest;
00033 BUTTON_STRUCT b_filetest;
00034 BUTTON_STRUCT b_our_tracking;
00035 BUTTON_STRUCT b_ref_tracking;
00036 BUTTON_STRUCT b_photo_mode;
00037
00038 static void b_our_tracking_cb(void* button)
00039 {
00040     tracking_set_mode(OUR_TRACKING);
00041     gui_screen_navigate(get_screen_tracking());
00042 }
00043
00044 static void b_ref_tracking_cb(void* button)
00045 {
00046     tracking_set_mode(REFERENCE_TRACKING);
00047     gui_screen_navigate(get_screen_tracking());
00048 }
00049
00050 static void b_photo_mode_cb(void* button)
00051 {
00052     gui_screen_navigate(get_screen_photomode());
00053 }
00054
00055 static void b_guitest_cb(void* button)
00056 {
00057     gui_screen_navigate(get_screen_guitest());
00058 }
00059
00060 static void b_filetest_cb(void* button)
00061 {
00062     gui_screen_navigate(get_screen_filetest());
00063 }
00064
00065 static void b_pixytest_cb(void* button)
00066 {
00067     gui_screen_navigate(get_screen_pixytest());
00068 }
00069
00070
00071 static void enter(void* screen)
00072 {
00073     tft_clear(WHITE);

```

```

00074
00075 //Heading
00076 tft_print_line(10, 10, BLUE, TRANSPARENT, 1, "Discoverpixy");
00077 tft_draw_line(0, 40, 319, 40, BLACK);
00078
00079 #define X_TAB          97
00080 #define BUTTON_SPACING 7
00081
00082 //First line of buttons
00083 #define Y_FIRST        60
00084 tft_print_line(10, Y_FIRST, BLACK, TRANSPARENT, 0, "Tracking:");
00085
00086 b_our_tracking.base.x1 = X_TAB; //Start X of Button
00087 b_our_tracking.base.y1 = Y_FIRST - 3; //Start Y of Button
00088 b_our_tracking.base.x2 = AUTO; //Auto Calculate X2 with String Width
00089 b_our_tracking.base.y2 = AUTO; //Auto Calculate Y2 with String Height
00090 b_our_tracking.txtcolor = WHITE; //Set foreground color
00091 b_our_tracking.bgcolor = HEX(0xE30535); //Set background color (Don't take 255 or 0 on at
least one channel, to make shadows possible)
00092 b_our_tracking.font = 0; //Select Font
00093 b_our_tracking.text = "Our Tracking"; //Set Text (For formatted strings take sprintf)
00094 b_our_tracking.callback = b_our_tracking_cb; //Call b_our_tracking when the
button get's pressed
00095 gui_button_add(&b_our_tracking); //Register Button (and run the callback from now on)
00096
00097
00098 b_ref_tracking.base.x1 = b_our_tracking.base.x2 + BUTTON_SPACING;
00099 b_ref_tracking.base.y1 = Y_FIRST - 3;
00100 b_ref_tracking.base.x2 = AUTO;
00101 b_ref_tracking.base.y2 = AUTO;
00102 b_ref_tracking.txtcolor = WHITE;
00103 b_ref_tracking.bgcolor = HEX(0xFF2151);
00104 b_ref_tracking.font = 0;
00105 b_ref_tracking.text = "Ref Tracking";
00106 b_ref_tracking.callback = b_ref_tracking_cb;
00107 gui_button_add(&b_ref_tracking);
00108
00109 //Second line of buttons
00110 #define Y_SECOND Y_FIRST+25
00111 tft_print_line(10, Y_SECOND, BLACK, TRANSPARENT, 0, "Photo mode:"
);
00112
00113 b_photo_mode.base.x1 = X_TAB;
00114 b_photo_mode.base.y1 = Y_SECOND - 3;
00115 b_photo_mode.base.x2 = AUTO;
00116 b_photo_mode.base.y2 = AUTO;
00117 b_photo_mode.txtcolor = WHITE;
00118 b_photo_mode.bgcolor = HEX(0x21B1FF);
00119 b_photo_mode.font = 0;
00120 b_photo_mode.text = "Photo Mode";
00121 b_photo_mode.callback = b_photo_mode_cb;
00122 gui_button_add(&b_photo_mode);
00123
00124
00125 //Third line of buttons
00126 #define Y_THIRD Y_SECOND+25
00127 tft_print_line(10, Y_THIRD, BLACK, TRANSPARENT, 0, "Tests:");
00128
00129 b_guittest.base.x1 = X_TAB;
00130 b_guittest.base.y1 = Y_THIRD - 3;
00131 b_guittest.base.x2 = AUTO;
00132 b_guittest.base.y2 = AUTO;
00133 b_guittest.txtcolor = BLACK;
00134 b_guittest.bgcolor = HEX(0x00FA21);
00135 b_guittest.font = 0;
00136 b_guittest.text = "Gui & Tft";
00137 b_guittest.callback = b_guittest_cb;
00138 gui_button_add(&b_guittest);
00139
00140
00141 b_pixytest.base.x1 = b_guittest.base.x2 + BUTTON_SPACING;
00142 b_pixytest.base.y1 = Y_THIRD - 3;
00143 b_pixytest.base.x2 = AUTO;
00144 b_pixytest.base.y2 = AUTO;
00145 b_pixytest.txtcolor = BLACK;
00146 b_pixytest.bgcolor = HEX(0x00FA96);
00147 b_pixytest.font = 0;
00148 b_pixytest.text = "Pixy";
00149 b_pixytest.callback = b_pixytest_cb;
00150 gui_button_add(&b_pixytest);
00151
00152
00153 b_filetest.base.x1 = b_pixytest.base.x2 + BUTTON_SPACING;
00154 b_filetest.base.y1 = Y_THIRD - 3;
00155 b_filetest.base.x2 = AUTO;
00156 b_filetest.base.y2 = AUTO;
00157 b_filetest.txtcolor = BLACK;

```

```

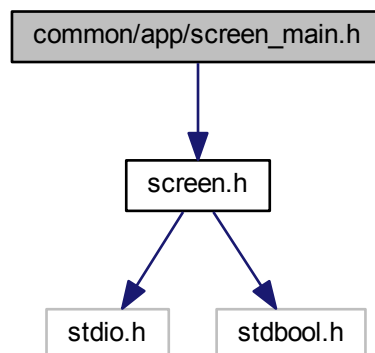
00158     b_filetest.bgcolor = HEX(0x00FAC4);
00159     b_filetest.font = 0;
00160     b_filetest.text = "File";
00161     b_filetest.callback = b_filetest_cb;
00162     gui_button_add(&b_filetest);
00163
00164
00165     //Bottom line
00166     tft_draw_line(0, 145, 319, 145, BLACK);
00167     tft_print_line(10, 150, BLUE, TRANSPARENT, 0, "Powered by");
00168     tft_draw_bitmap_file_unscaled(10, 165, "pixy_small.bmp");
00169     tft_draw_bitmap_file_unscaled(165, 165, "stm_small.bmp");
00170
00171 }
00172
00173 static void leave(void* screen)
00174 {
00175     gui_button_remove(&b_our_tracking);
00176     gui_button_remove(&b_ref_tracking);
00177     gui_button_remove(&b_photo_mode);
00178     gui_button_remove(&b_guitest);
00179     gui_button_remove(&b_pixytest);
00180     gui_button_remove(&b_filetest);
00181 }
00182
00183 static void update(void* screen)
00184 {
00185     //gui_button_redraw(&b_guitest); //only needed if button is overdrawn by others
00186 }
00187
00188
00189 static SCREEN_STRUCT screen = {
00190     enter,
00191     leave,
00192     update
00193 };
00194
00195
00196 SCREEN_STRUCT* get_screen_main()
00197 {
00198     return &screen;
00199 }

```

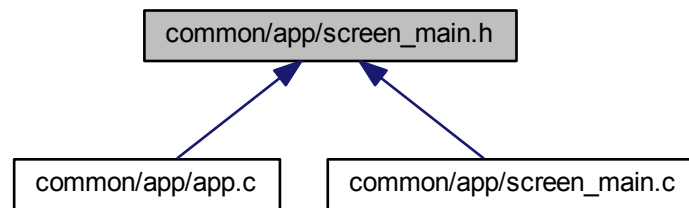
7.23 common/app/screen_main.h File Reference

```
#include "screen.h"
```

Include dependency graph for screen_main.h:



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



Functions

- `SCREEN_STRUCT * get_screen_main ()`

7.24 screen_main.h

```

00001 /*****
00002 * Project:      discoverpixy
00003 * Website:     https://github.com/t-moe/discoverpixy
00004 * Authors:     Aaron Schmocker, Timo Lang
00005 * Institution: BFH Bern University of Applied Sciences
00006 * File:       common/app/screen_main.h
00007 *
00008 * Version History:
00009 * Date      Autor Email      SHA      Changes
00010 * 2015-04-27 timolang@gmail.com cf72baa Introduced a Screen (sub) module and divided app into multiple
00011 * 2015-05-10 timolang@gmail.com 21edc56 Added doxyfile (doxygen) for the common folder. Started with
00012 * 2015-05-11 timolang@gmail.com 08d9fe0 More work on doxygen module structure
00013 * 2015-05-12 timolang@gmail.com 1402598 Added doxygen stuff for button module and some minor changes to
00014 * 2015-05-15 timolang@gmail.com 9a16865 Added doxygen comments to filesystem, checkbox, numupdown and
00015 * screen module. And some minor changes to the other modules.
00016 *****/
00017
00018 #include "screen.h"
00019
00024
00030
00031
00036
00042
00043
00049 SCREEN_STRUCT* get_screen_main();
00050

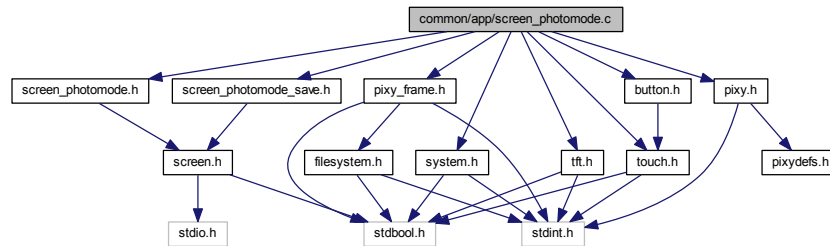
```

7.25 common/app/screen_photomode.c File Reference

```
#include "screen_photomode.h"
```

```
#include "screen_photomode_save.h"
#include "button.h"
#include "tft.h"
#include "touch.h"
#include "pixy.h"
#include "system.h"
#include "pixy_frame.h"
```

Include dependency graph for screen_photomode.c:



Functions

- static void [b_back_cb](#) (void *button)
- static void [b_save_cb](#) (void *button)
- static void [touchCB](#) (void *touchArea, [TOUCH_ACTION](#) triggeredAction)
- static void [enter](#) (void *screen)
- static void [leave](#) (void *screen)
- static void [update](#) (void *screen)
- [SCREEN_STRUCT](#) * [get_screen_photomode](#) ()

Variables

- static bool [pixy_connected](#) = false
- static [BUTTON_STRUCT](#) [b_back](#)
- static [BUTTON_STRUCT](#) [b_save](#)
- static [TOUCH_AREA_STRUCT](#) [a_area](#)
- static bool [subMenu](#) = false
- static [POINT_STRUCT](#) [pixy_pos](#)
- static [POINT_STRUCT](#) [old_pos](#)
- static [SCREEN_STRUCT](#) [screen](#)

7.25.1 Function Documentation

7.25.1.1 static void [b_back_cb](#) (void * [button](#)) [static]

Definition at line 33 of file [screen_photomode.c](#).

```
00034 {
00035     subMenu = false; //we're not entering a submenu
00036     gui_screen_back(); //navigate back to the previous screen
00037 }
```

Here is the call graph for this function:



Here is the caller graph for this function:

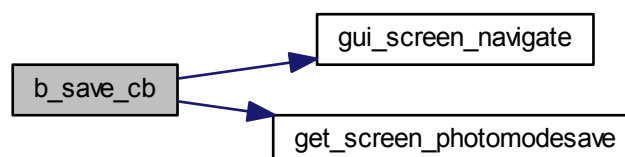


7.25.1.2 static void b_save_cb (void * *button*) [static]

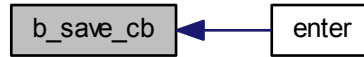
Definition at line 40 of file [screen_photomode.c](#).

```
00041 {  
00042     subMenu = true; //we're entering a submenu  
00043     gui_screen_navigate(get_screen_photomodesave()); //navigate  
        to the save screen  
00044 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



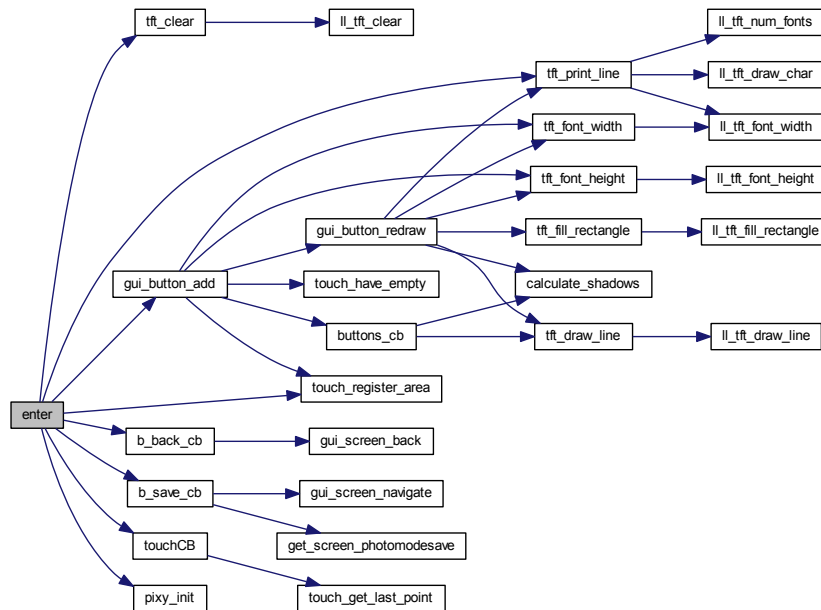
7.25.1.3 static void enter (void * screen) [static]

Definition at line 107 of file [screen_photomode.c](#).

```

00108 {
00109     tft_clear(WHITE);
00110
00111     tft_print_line(5, 5, BLACK, TRANSPARENT, 0, "Drag the image around and ")
00112 ;
00113     //Back button
00114     b_back.base.x1 = 5; //Start X of Button
00115     b_back.base.y1 = 19; //Start Y of Button
00116     b_back.base.x2 = AUTO; //Auto Calculate X2 with String Width
00117     b_back.base.y2 = AUTO; //Auto Calculate Y2 with String Height
00118     b_back.txtcolor = WHITE; //Set foreground color
00119     b_back.bgcolor = HEX(0xAE1010); //Set background color (Don't take 255 or 0 on at least
one channel, to make shadows possible)
00120     b_back.font = 0; //Select Font
00121     b_back.text = "Back"; //Set Text (For formatted strings take sprintf)
00122     b_back.callback = b_back_cb; //Call b_back_cb as Callback
00123     gui_button_add(&b_back); //Register Button (and run the callback from now on)
00124
00125     //Save button
00126     b_save.base.x1 = 190;
00127     b_save.base.y1 = 3;
00128     b_save.base.x2 = AUTO;
00129     b_save.base.y2 = AUTO;
00130     b_save.txtcolor = WHITE;
00131     b_save.bgcolor = HEX(0x1010AE);
00132     b_save.font = 0;
00133     b_save.text = "Save it!";
00134     b_save.callback = b_save_cb;
00135     gui_button_add(&b_save);
00136
00137     //Frame Coordinates: topleft = (1,40); bottomright = (318,238)
00138     //Leave a 10px border for the area
00139
00140     //Area to drag the image around
00141     a_area.hookedActions = PEN_DOWN | PEN_MOVE |
PEN_ENTER | PEN_UP | PEN_LEAVE;
00142     a_area.x1 = 11;
00143     a_area.y1 = 50;
00144     a_area.x2 = 308;
00145     a_area.y2 = 228;
00146     a_area.callback = touchCB;
00147     touch_register_area(&a_area);
00148
00149     //Pixy stuff
00150     pixy_connected = (pixy_init() == 0); //try to connect to pixy
00151
00152     if (pixy_connected && !subMenu) { //pixy is connected, but we are not coming from
a submenu
00153         pixy_pos.x = pixy_pos.y = 500; //reset servo positions to center
00154     }
00155 }
  
```

Here is the call graph for this function:



7.25.1.4 static void leave (void * *screen*) [static]

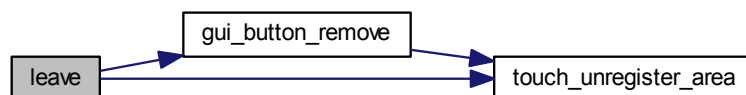
Definition at line 158 of file [screen_photomode.c](#).

```

00159 {
00160     //remove buttons and touch area.
00161     gui_button_remove(&b_back);
00162     gui_button_remove(&b_save);
00163     touch_unregister_area(&a_area);
00164 }

```

Here is the call graph for this function:



7.25.1.5 static void touchCB (void * *touchArea*, TOUCH_ACTION *triggeredAction*) [static]

Definition at line 50 of file [screen_photomode.c](#).

```

00051 {
00052     POINT_STRUCT p = touch_get_last_point(); //get the last touched point
00053 }

```



```

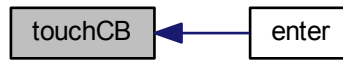
00054     switch (triggeredAction) {
00055     case PEN_ENTER:
00056     case PEN_DOWN:
00057         old_pos = p; //If the user "newly" enters the touch area, we set the "last" position to the
current
00058         break;
00059
00060     case PEN_MOVE: { //the user is moving around, he entered the screen a while ago (old_pos is
set)
00061         int16_t deltaX = p.x - old_pos.x; //Calculate x difference between last and current touch
00062         int16_t deltaY = p.y - old_pos.y; //Calculate y difference between last and current touch
00063         old_pos = p; //store the current touch point for the next time
00064
00065         //printf("%d %d\n",deltaX,deltaY);
00066         if (pixy_connected) {
00067             //Calculate new servo coordinates. 2 is just a proportional factor
00068             int16_t new_x = pixy_pos.x + deltaX * 2;
00069             int16_t new_y = pixy_pos.y - deltaY * 2;
00070
00071             //check limits
00072             if (new_x < 0) {
00073                 new_x = 0;
00074             }
00075
00076             if (new_x > 1000) {
00077                 new_x = 1000;
00078             }
00079
00080             if (new_y < 0) {
00081                 new_y = 0;
00082             }
00083
00084             if (new_y > 1000) {
00085                 new_y = 1000;
00086             }
00087
00088             //set pixy_pos so that the main routine can send it to the servos
00089             pixy_pos.x = new_x;
00090             pixy_pos.y = new_y;
00091         }
00092     }
00093     break;
00094
00095     case PEN_UP:
00096     case PEN_LEAVE:
00097         //printf("Leave/up\n");
00098         break;
00099
00100     default:
00101         break;
00102     }
00103
00104 }

```

Here is the call graph for this function:



Here is the caller graph for this function:



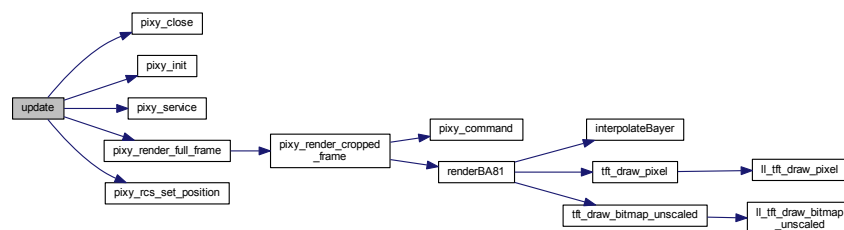
7.25.1.6 static void update (void * screen) [static]

Definition at line 168 of file [screen_photomode.c](#).

```

00169 {
00170     //Note: The only way to detect that pixy has been disconnected is if a command fails. There's no
00171     pixy_is_connected method yet :(
00172     if (!pixy_connected) { //Pixy not connected
00173         pixy_close(); //Ensure that all pixy resources are freed (failsafe)
00174     }
00175     if (pixy_init() == 0) { //try to connect to pixy
00176         pixy_connected = true;
00177     }
00178     if (!subMenu) { //we're not coming from a submenu
00179         pixy_pos.x = pixy_pos.y = 500; //reset servo positions to center
00180     }
00181     printf("pixy (re)initialized\n");
00182 }
00183
00184
00185
00186 if (pixy_connected) { //If we are connected (now)
00187     pixy_service(); //Handle pending pixy events (e.g. color info retrieval)
00188     pixy_render_full_frame(1, 40); //render the pixy video at point (1,40)
00189
00190     //set the servo positions to the coordinates from the touch interrupt
00191     pixy_rcs_set_position(0, pixy_pos.x);
00192     pixy_rcs_set_position(1, pixy_pos.y);
00193 }
00194 }
00195 }
  
```

Here is the call graph for this function:



7.25.2 Variable Documentation

7.25.2.1 TOUCH_AREA_STRUCT a_area [static]

Definition at line 29 of file [screen_photomode.c](#).

7.25.2.2 `BUTTON_STRUCT b_back` `[static]`

Definition at line 27 of file [screen_photomode.c](#).

7.25.2.3 `BUTTON_STRUCT b_save` `[static]`

Definition at line 28 of file [screen_photomode.c](#).

7.25.2.4 `POINT_STRUCT old_pos` `[static]`

Definition at line 47 of file [screen_photomode.c](#).

7.25.2.5 `bool pixy_connected = false` `[static]`

Definition at line 25 of file [screen_photomode.c](#).

7.25.2.6 `POINT_STRUCT pixy_pos` `[static]`

Definition at line 46 of file [screen_photomode.c](#).

7.25.2.7 `SCREEN_STRUCT screen` `[static]`

Initial value:

```
= {
    enter,
    leave,
    update
}
```

Definition at line 198 of file [screen_photomode.c](#).

7.25.2.8 `bool subMenu = false` `[static]`

Definition at line 30 of file [screen_photomode.c](#).

7.26 `screen_photomode.c`

```
00001 /*****
00002  * Project:      discoverpixy
00003  * Website:     https://github.com/t-moe/discoverpixy
00004  * Authors:     Aaron Schmocker, Timo Lang
00005  * Institution: BFH Bern University of Applied Sciences
00006  * File:       common/app/screen_photomode.c
00007  *
00008  * Version History:
00009  * Date      Autor Email      SHA      Changes
00010  * 2015-05-15 timolang@gmail.com 27c09ba Redesigned main menu. Moved stuff from pixytest to a new helper
00011  * 2015-05-16 timolang@gmail.com 62006e0 Documented pixy_helper and implemented/finished photo-mode
00012  * 2015-06-07 timolang@gmail.com c87220d Renamed pixy_helper to pixy_frame. Updated docu of appliaction.
00013  * added doxygen comments to pixy_{frame,control}.h
00014  *
00015  *****/
00016 #include "screen_photomode.h"
00017 #include "screen_photomode_save.h"
00018 #include "button.h"
```

```

00019 #include "tft.h"
00020 #include "touch.h"
00021 #include "pixy.h"
00022 #include "system.h"
00023 #include "pixy_frame.h"
00024
00025 static bool pixy_connected = false; //Whether or not the pixy cam is currently connected
00026
00027 static BUTTON_STRUCT b_back; //Button to navigate back
00028 static BUTTON_STRUCT b_save; //Button to save the current image
00029 static TOUCH_AREA_STRUCT a_area; //Touch Area, where the frame is drawn. Used to
    drag the image around
00030 static bool subMenu = false; //Whether or not we left the current screen for a submenu
00031
00032 //Callback for when the user presses the "back" button
00033 static void b_back_cb(void* button)
00034 {
00035     subMenu = false; //we're not entering a submenu
00036     gui_screen_back(); //navigate back to the previous screen
00037 }
00038
00039 //Callback for when the user presses the "save" button
00040 static void b_save_cb(void* button)
00041 {
00042     subMenu = true; //we're entering a submenu
00043     gui_screen_navigate(get_screen_photomodesave()); //navigate
    to the save screen
00044 }
00045
00046 static POINT_STRUCT pixy_pos; //The current position of pixy's servos
00047 static POINT_STRUCT old_pos; //The last touch position on the screen
00048
00049 //Callback for when the user drags the image around
00050 static void touchCB(void* touchArea, TOUCH_ACTION triggeredAction)
00051 {
00052     POINT_STRUCT p = touch_get_last_point(); //get the last touched point
00053
00054     switch (triggeredAction) {
00055     case PEN_ENTER:
00056     case PEN_DOWN:
00057         old_pos = p; //If the user "newly" enters the touch area, we set the "last" position to the current
00058         break;
00059
00060     case PEN_MOVE: { //the user is moving around, he entered the screen a while ago (old_pos is
    set)
00061         int16_t deltaX = p.x - old_pos.x; //Calculate x difference between last and current touch
00062         int16_t deltaY = p.y - old_pos.y; //Calculate y difference between last and current touch
00063         old_pos = p; //store the current touch point for the next time
00064
00065         //printf("%d %d\n",deltaX,deltaY);
00066         if (pixy_connected) {
00067             //Calculate new servo coordinates. 2 is just a proportional factor
00068             int16_t new_x = pixy_pos.x + deltaX * 2;
00069             int16_t new_y = pixy_pos.y - deltaY * 2;
00070
00071             //check limits
00072             if (new_x < 0) {
00073                 new_x = 0;
00074             }
00075
00076             if (new_x > 1000) {
00077                 new_x = 1000;
00078             }
00079
00080             if (new_y < 0) {
00081                 new_y = 0;
00082             }
00083
00084             if (new_y > 1000) {
00085                 new_y = 1000;
00086             }
00087
00088             //set pixy_pos so that the main routine can send it to the servos
00089             pixy_pos.x = new_x;
00090             pixy_pos.y = new_y;
00091         }
00092     }
00093     break;
00094
00095     case PEN_UP:
00096     case PEN_LEAVE:
00097         //printf("Leave/up\n");
00098         break;
00099
00100     default:
00101         break;
00102 }

```

```

00103
00104 }
00105
00106 //Callback for when the screen is entered/loaded
00107 static void enter(void* screen)
00108 {
00109     tft_clear(WHITE);
00110
00111     tft_print_line(5, 5, BLACK, TRANSPARENT, 0, "Drag the image around and ")
00112 ;
00113
00114     //Back button
00115     b_back.base.x1 = 5; //Start X of Button
00116     b_back.base.y1 = 19; //Start Y of Button
00117     b_back.base.x2 = AUTO; //Auto Calculate X2 with String Width
00118     b_back.base.y2 = AUTO; //Auto Calculate Y2 with String Height
00119     b_back.txtcolor = WHITE; //Set foreground color
00120     b_back.bgcolor = HEX(0xAE1010); //Set background color (Don't take 255 or 0 on at least one
channel, to make shadows possible)
00121     b_back.font = 0; //Select Font
00122     b_back.text = "Back"; //Set Text (For formatted strings take sprintf)
00123     b_back.callback = b_back_cb; //Call b_back_cb as Callback
00124     gui_button_add(&b_back); //Register Button (and run the callback from now on)
00125
00126     //Save button
00127     b_save.base.x1 = 190;
00128     b_save.base.y1 = 3;
00129     b_save.base.x2 = AUTO;
00130     b_save.base.y2 = AUTO;
00131     b_save.txtcolor = WHITE;
00132     b_save.bgcolor = HEX(0x1010AE);
00133     b_save.font = 0;
00134     b_save.text = "Save it!";
00135     b_save.callback = b_save_cb;
00136     gui_button_add(&b_save);
00137
00138     //Frame Coordinates: topleft = (1,40); bottomright = (318,238)
00139     //Leave a 10px border for the area
00140
00141     //Area to drag the image around
00142     a_area.hookedActions = PEN_DOWN | PEN_MOVE |
PEN_ENTER | PEN_UP | PEN_LEAVE;
00143     a_area.x1 = 11;
00144     a_area.y1 = 50;
00145     a_area.x2 = 308;
00146     a_area.y2 = 228;
00147     a_area.callback = touchCB;
00148     touch_register_area(&a_area);
00149
00150     //Pixy stuff
00151     pixy_connected = (pixy_init() == 0); //try to connect to pixy
00152
00153     if (pixy_connected && !subMenu) { //pixy is connected, but we are not coming from
a submenu
00154         pixy_pos.x = pixy_pos.y = 500; //reset servo positions to center
00155     }
00156
00157 //Callback for when the screen is left/unloaded
00158 static void leave(void* screen)
00159 {
00160     //remove buttons and touch area.
00161     gui_button_remove(&b_back);
00162     gui_button_remove(&b_save);
00163     touch_unregister_area(&a_area);
00164 }
00165
00166 //Callback for when the screen should be updated
00167 //This is the main loop of the screen. This method will be called repeatedly
00168 static void update(void* screen)
00169 {
00170     //Note: The only way to detect that pixy has been disconnected is if a command fails. There's no
pixy_is_connected method yet :(
00171
00172     if (!pixy_connected) { //Pixy not connected
00173         pixy_close(); //Ensure that all pixy resources are freed (failsafe)
00174
00175         if (pixy_init() == 0) { //try to connect to pixy
00176             pixy_connected = true;
00177
00178             if (!subMenu) { //we're not coming from a submenu
00179                 pixy_pos.x = pixy_pos.y = 500; //reset servo positions to center
00180             }
00181
00182             printf("pixy (re)initialized\n");
00183         }
00184     }

```

```

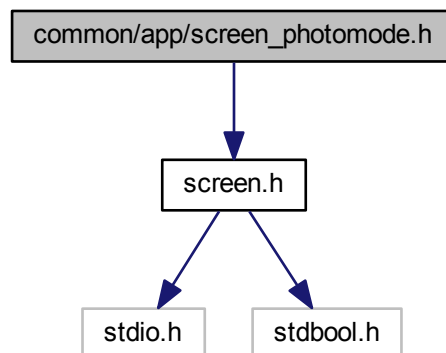
00185
00186     if (pixy_connected) { //If we are connected (now)
00187         pixy_service(); //Handle pending pixy events (e.g. color info retrieval)
00188
00189         pixy_render_full_frame(1, 40); //render the pixy video at point (1,40)
00190
00191         //set the servo positions to the coordinates form the touch interrupt
00192         pixy_rcs_set_position(0, pixy_pos.x);
00193         pixy_rcs_set_position(1, pixy_pos.y);
00194     }
00195 }
00196
00197 //Declare screen callbacks
00198 static SCREEN_STRUCT screen = {
00199     enter,
00200     leave,
00201     update
00202 };
00203
00204
00205 SCREEN_STRUCT* get_screen_photomode()
00206 {
00207     return &screen;
00208 }

```

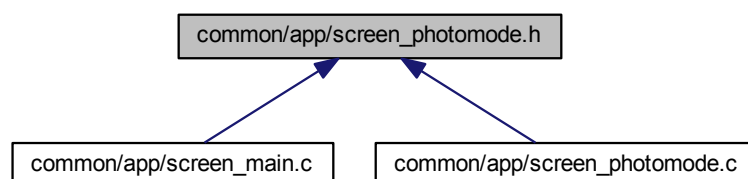
7.27 common/app/screen_photomode.h File Reference

```
#include "screen.h"
```

Include dependency graph for screen_photomode.h:



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



Functions

- `SCREEN_STRUCT * get_screen_photomode ()`

7.28 screen_photomode.h

```

00001 /*****
00002  * Project:      discoverpixy
00003  * Website:     https://github.com/t-moe/discoverpixy
00004  * Authors:     Aaron Schmocker, Timo Lang
00005  * Institution: BFH Bern University of Applied Sciences
00006  * File:        common/app/screen_photomode.h
00007  *
00008  * Version History:
00009  * Date          Autor Email          SHA          Changes
00010  * 2015-05-15    timolang@gmail.com    27c09ba    Redesigned main menu. Moved stuff from pixytest to a new helper
00011  *               file and to the new "photo mode"-screen.
00012  *****/
00013
00014 #include "screen.h"
00015
00020
00026
00032 SCREEN_STRUCT* get_screen_photomode ();
00033

```

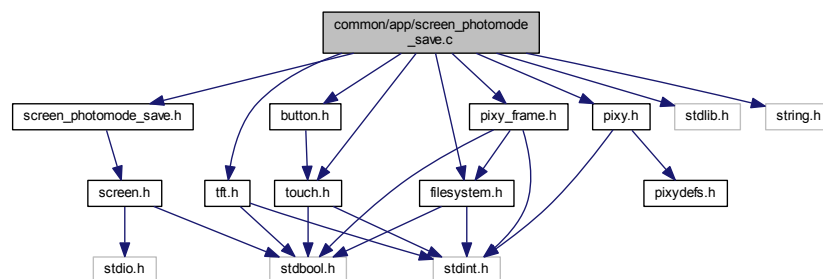
7.29 common/app/screen_photomode_save.c File Reference

```

#include "screen_photomode_save.h"
#include "filesystem.h"
#include "button.h"
#include "tft.h"
#include "touch.h"
#include "pixy.h"
#include "pixy_frame.h"
#include <stdlib.h>
#include <string.h>

```

Include dependency graph for screen_photomode_save.c:



Data Structures

- struct `FILE_LIST_ENTRY_S`

Macros

- `#define X_OFS 5`

Typedefs

- `typedef struct FILE_LIST_ENTRY_S FILE_LIST_ENTRY`

Enumerations

- `enum {
 init, error, showlist, picking,
 saving, done }`

Functions

- `static void b_back_cb (void *button)`
- `static void touchCB (void *touchArea, TOUCH_ACTION triggeredAction)`
- `static void enter (void *screen)`
- `static void update (void *screen)`
- `static void leave (void *screen)`
- `SCREEN_STRUCT * get_screen_photomodesave ()`

Variables

- `static BUTTON_STRUCT b_back`
- `static TOUCH_AREA_STRUCT a_area`
- `static int num_files_ok`
- `static enum { ... } state`
- `static int fontheight`
- `static int liststart`
- `static const char * picked_file`
- `static FILE_LIST_ENTRY * files_ok`
- `static const char * nomatch_text []`
- `static unsigned char bmpheader_data [0x7A]`
- `static SCREEN_STRUCT screen`

7.29.1 Macro Definition Documentation

7.29.1.1 `#define X_OFS 5`

7.29.2 Typedef Documentation

7.29.2.1 `typedef struct FILE_LIST_ENTRY_S FILE_LIST_ENTRY`

7.29.3 Enumeration Type Documentation

7.29.3.1 anonymous enum

Enumerator

init

error

showlist
picking
saving
done

Definition at line 36 of file [screen_photomode_save.c](#).

```
00036 {init, error, showlist, picking, saving, done}
      state; //Current state of the screen state machine
```

7.29.4 Function Documentation

7.29.4.1 static void b_back_cb (void * button) [static]

Definition at line 30 of file [screen_photomode_save.c](#).

```
00031 {
00032     gui_screen_back();
00033 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



7.29.4.2 static void enter (void * screen) [static]

Definition at line 108 of file [screen_photomode_save.c](#).

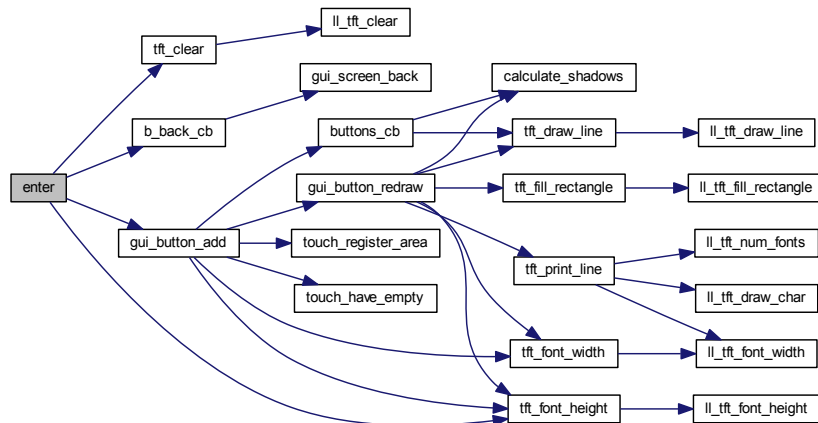
```
00109 {
00110     tft_clear(WHITE);
00111
00112
00113     #define X_OFS 5
00114
00115     //Back button
00116     b_back.base.x1 = X_OFS; //Start X of Button
00117     b_back.base.y1 = 210; //Start Y of Button
```

```

00118     b_back.base.x2 = AUTO; //Auto Calculate X2 with String Width
00119     b_back.base.y2 = AUTO; //Auto Calculate Y2 with String Height
00120     b_back.txtcolor = WHITE; //Set foreground color
00121     b_back.bgcolor = HEX(0xAE1010); //Set background color (Don't take 255 or 0 on at least
one channel, to make shadows possible)
00122     b_back.font = 0; //Select Font
00123     b_back.text = "Back"; //Set Text (For formatted strings take sprintf)
00124     b_back.callback = b_back_cb; //Call b_back_cb as Callback
00125     gui_button_add(&b_back); //Register Button (and run the callback from now on)
00126
00127     state = init; //Start with the init state
00128     fontheight = tft_font_height(0) + 2; //Save the height of the used font, for
fast access
00129     files_ok = NULL; //initialize the linked list with 0 elements
00130     num_files_ok = 0; //we have zero! elements
00131 }

```

Here is the call graph for this function:



7.29.4.3 static void leave (void * screen) [static]

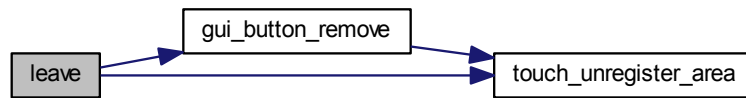
Definition at line 310 of file [screen_photomode_save.c](#).

```

00311 {
00312     gui_button_remove(&b_back); //Remove/Free the back button
00313
00314     if (state == picking) { //The user left the screen in the "picking"-phase
00315         touch_unregister_area(&a_area); //remove the touch area (for the list)
00316     }
00317
00318     if (state == picking || state == saving || state ==
done) { //the user left the screen after we created the linked list
00319         //Iterate through the linked list and free all resources
00320         FILE_LIST_ENTRY* current_entry = files_ok; //start with the list head
00321
00322         while (current_entry != NULL) { //while we're not at the end
00323             FILE_LIST_ENTRY* temp = current_entry->next; //save the next pointer because
we free the current element on the next line
00324             free((void*) (current_entry->filename)); //free filename
00325             free(current_entry); //free element itself
00326             current_entry = temp; //advance
00327         }
00328     }
00329 }

```

Here is the call graph for this function:



7.29.4.4 static void touchCB (void * touchArea, TOUCH_ACTION triggeredAction) [static]

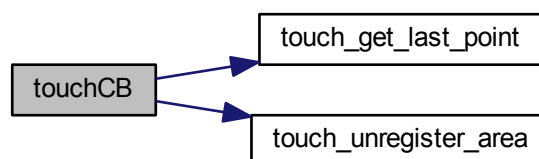
Definition at line 50 of file [screen_photomode_save.c](#).

```

00051 {
00052
00053     int y = touch_get_last_point().y - liststart; //Calculate the
y-Coordinate of the touch point relative to the start of the file-list
00054     int elem = y / fontheight; //Calculate the file index
00055
00056     if (elem < 0 | elem >= num_files_ok) {
00057         return; //Check if the file index is valid (0,1,..,num_files_ok-1)
00058     }
00059
00060     //Search for the corresponding entry in the linked list
00061     FILE_LIST_ENTRY* current_entry = files_ok; //Start walking through the list,
starting by the head of the list
00062
00063     for (int i = 0; i < elem; i++) { //Until we have reached the file (index)
00064         current_entry = current_entry->next; //traverse to the next file
00065     }
00066
00067     picked_file = current_entry->filename; //save the picked filename. It will be used
by the statemachine in the main loop
00068     touch_unregister_area(&a_area); //unregister the touch area, we no longer
need it. No more interrupts will be fired.
00069     state = saving; //Change the state of the statemachine
00070 }

```

Here is the call graph for this function:



Here is the caller graph for this function:



7.29.4.5 static void update (void * screen) [static]

Definition at line 135 of file `screen_photomode_save.c`.

```

00136 {
00137     switch (state) {
00138     case init: { //Init State: The user just entered the screen
00139         DIRECTORY_STRUCT* dir = filesystem_dir_open("."); //open root
        directory
00140
00141         if (dir == NULL) { //error while opening root directory
00142             tft_print_line(X_OFS, 5, BLACK, TRANSPARENT, 0, "Error
        accessing Filesystem");
00143             state = error;
00144             break;
00145         }
00146
00147         bool nomatch = true; //whether or not we have zero files which are suitable for saving
00148
00149         for (int i = 0; i < dir->num_files; i++) { //walk through all files in the directory
00150             FILE_STRUCT* file = &(dir->files[i]); //Pointer to the current
        file/subdirectory
00151
00152             //Ignore directories, archives, hidden files, system files and files we cannot write to
00153             if (file->fattrib & (F_SYS | F_HID | F_ARC |
        F_DIR | F_RDO)) {
00154                 continue;
00155             }
00156
00157             //ignore files which are not large enough
00158             if (file->fsize < 189410) {
00159                 continue; //size taken from an example bitmap (318x198x24)
00160             }
00161
00162             nomatch = false; //at least one file matches
00163             break;
00164         }
00165
00166         if (nomatch) { //not one file is suitable for writing
00167             int y = 5; //y-Coordinate where to start writing the error text
00168             int i = 0;
00169
00170             while (nomatch_text[i] != NULL) { //for every line in the big error array
00171                 //Write the line's text and go to the next line
00172                 tft_print_line(X_OFS, y + i * fontheight,
        BLACK, TRANSPARENT, 0, nomatch_text[i]);
00173                 i++;
00174             }
00175
00176             state = error;
00177         } else { //we have a least one suitable file
00178             state = showlist;
00179         }
00180
00181         filesystem_dir_close(dir); //free directory struct
00182     }
00183     break;
00184
00185     case showlist: { //Show List State: Where we load and present the suitable file's to the user
        in a list
00186         DIRECTORY_STRUCT* dir2 = filesystem_dir_open("."); //Open the
        directory again
00187
00188         if (dir2 == NULL) {
  
```

```

00189         return;    //Error on opening? This should never happen, since it's handled in the previous
state
00190     }
00191
00192     int y = 5; //y-Coordinate where to start drawing/writing text/list-elements
00193
00194     tft_print_line(X_OFS, y, BLACK, TRANSPARENT, 0, "Pick a file to
save the image to");
00195     y += fontheight + 5;
00196
00197     tft_print_line(X_OFS, y, BLUE, TRANSPARENT, 0, "Name
Modified
Size");
00198     y += fontheight;
00199
00200     liststart = y; //store the y coordinate of the start of the list away (used in toucharea
callback)
00201     num_files_ok = 0; //we start with 0 matching files
00202
00203     FILE_LIST_ENTRY* current_entry = NULL; //We start with an empty list
00204
00205     for (int i = 0; i < dir2->num_files && num_files_ok < 10; i++) { //go through
all the files of the directory, abort if we have 10 matches
00206         FILE_STRUCT* file = &(dir2->files[i]);
00207
00208         //Ignore directories, archives, hidden files, system files and files we cannot write to
00209         if (file->fattrib & (F_SYS | F_HID | F_ARC |
F_DIR | F_RDO)) {
00210             continue;
00211         }
00212
00213         //ignore files which are not large enough
00214         if (file->fsize < 189410) {
00215             continue; //size taken from an example bitmap (318x198x24)
00216         }
00217
00218         //Print out filename, modified date,time and file size
00219         tft_print_formatted(X_OFS, y, BLACK,
TRANSPARENT, 0, "%-16s %02u.%02u.%02u %02u:%02u:%02u %u",
00220             file->fname,
00221             file->fdate.day,
00222             file->fdate.month,
00223             (file->fdate.year + 1980) % 100,
00224             file->ftime.hour,
00225             file->ftime.min,
00226             file->ftime.sec * 2,
00227             file->fsize);
00228
00229         if (current_entry == NULL) { //The list is empty
00230             current_entry = malloc(sizeof(FILE_LIST_ENTRY)); //create new entry
00231             files_ok = current_entry; //assign it to the list head
00232         } else { //there's a least one entry in the list
00233             current_entry->next = malloc(sizeof(FILE_LIST_ENTRY)); //append entry to
previous entry
00234             current_entry = current_entry->next; //newly created entry is the current now.
00235         }
00236
00237         current_entry->next = NULL; //we're at the end of the list (for now)
00238         current_entry->filename = malloc(strlen(file->fname) + 1); //allocate space for
the filename + zero-termination
00239         strcpy(current_entry->filename, file->fname); //copy filename (so that we can
close the directory after scanning)
00240
00241         //since we have found a suitable file we need to increment the position in the list
00242         num_files_ok++;
00243         y += fontheight;
00244     }
00245
00246     //Touch area for file-selection (in the list)
00247     a_area.hookedActions = PEN_UP; //we're only interested in PEN_UP events
00248     a_area.x1 = X_OFS; //Left border
00249     a_area.y1 = liststart; //Start where the list started
00250     a_area.x2 = 320 - X_OFS; //Right border
00251     a_area.y2 = liststart + fontheight *
num_files_ok; //stop at the end of the list
00252     a_area.callback = touchCB; //execute our callback when PEN_UP occurs
00253     touch_register_area(&a_area); //register the touch area and receive events
from now on
00254
00255     filesystem_dir_close(dir2); //we no longer need the directory struct, since we
have our own linked list now
00256
00257     state = picking;
00258 }
00259 break;
00260
00261 case picking: //Picking State: Where we wait on the users file choice
00262     pixy_service(); //Handle pending pixy events
00263

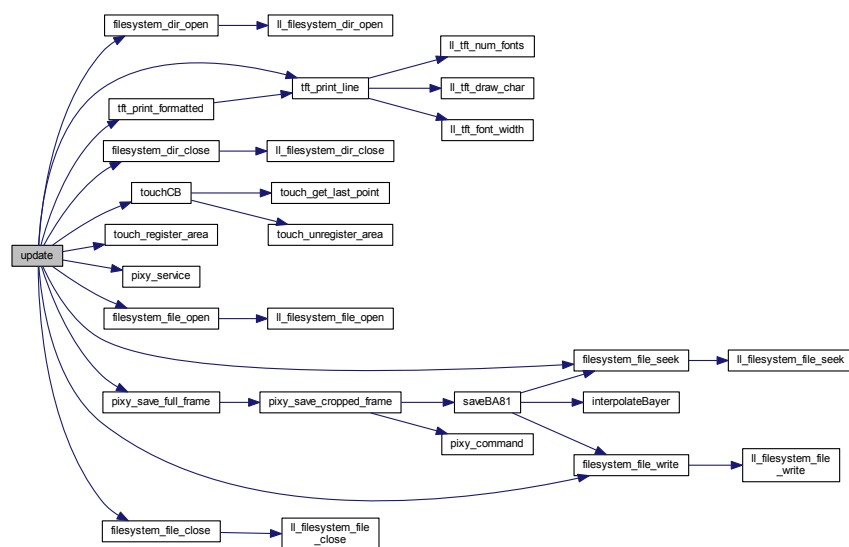
```

```

00264         //do nothing and wait on user to pick a file
00265         break;
00266
00267     case saving: { //Saving State: Where we save the image to the selected file
00268         FILE_HANDLE* file = filesystem_file_open(
00269             picked_file); //try to open the selected file
00270
00271         if (file == NULL) { //opening the file failed
00272             tft_print_formatted(X_OFS, 190, BLUE,
00273                 TRANSPARENT, 0, "Could not open %s", picked_file);
00274             state = error;
00275             break;
00276         }
00277         filesystem_file_seek(file, 0); //seek to the start of the file (optional?)
00278
00279         if (filesystem_file_write(file, bmpheader_data, 0x7A) !=
00280             F_OK) { //Writing the header failed
00281             tft_print_formatted(X_OFS, 190, BLUE,
00282                 TRANSPARENT, 0, "Error while writing to %s", picked_file);
00283             filesystem_file_close(file);
00284             state = error;
00285             break;
00286         }
00287         if (pixy_save_full_frame(file) != 0) { //Writing the imagedata failed
00288             tft_print_formatted(X_OFS, 190, BLUE,
00289                 TRANSPARENT, 0, "Error while writing to %s", picked_file);
00290             filesystem_file_close(file);
00291             state = error;
00292             break;
00293         }
00294         //if we reach this point, we have written all data out successfully
00295         filesystem_file_close(file); //close/finalize the file
00296         tft_print_formatted(X_OFS, 190, BLUE,
00297             TRANSPARENT, 0, "Image saved to %s", picked_file);
00298         state = done;
00299     }
00300     case error: //Error State: Where we show an error message and leave the user no other choice than
00301         to click the backbutton
00302     case done: //Done State: When saving the file was successful
00303         pixy_service(); //Handle pending pixy events
00304         //wait on user to click the back button
00305         break;
00306     }
00307 }

```

Here is the call graph for this function:



7.29.5 Variable Documentation

7.29.5.1 TOUCH_AREA_STRUCT a_area [static]

Definition at line 27 of file [screen_photomode_save.c](#).

7.29.5.2 BUTTON_STRUCT b_back [static]

Definition at line 26 of file [screen_photomode_save.c](#).

7.29.5.3 unsigned char bmpheader_data[0x7A] [static]

Initial value:

```
= {
    0x42, 0x4d, 0xe2, 0xe3, 0x02, 0x00, 0x00, 0x00, 0x00, 0x00, 0x7a, 0x00,
    0x00, 0x00, 0x6c, 0x00, 0x00, 0x00, 0x3e, 0x01, 0x00, 0x00, 0xc6, 0x00,
    0x00, 0x00, 0x01, 0x00, 0x18, 0x00, 0x00, 0x00, 0x00, 0x00, 0x68, 0xe3,
    0x02, 0x00, 0x13, 0x0b, 0x00, 0x00, 0x13, 0x0b, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x42, 0x47, 0x52, 0x73, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x02, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00
}
```

Definition at line 93 of file [screen_photomode_save.c](#).

7.29.5.4 FILE_LIST_ENTRY* files_ok [static]

Definition at line 47 of file [screen_photomode_save.c](#).

7.29.5.5 int fontheight [static]

Definition at line 37 of file [screen_photomode_save.c](#).

7.29.5.6 int liststart [static]

Definition at line 38 of file [screen_photomode_save.c](#).

7.29.5.7 const char* nomatch_text[] [static]

Initial value:

```
= {
    "Due to limitations of the filesystem",
    "implementation you can only write to",
    "existing files.",
    "",
    "The files need to have a .bmp",
    "extension and must be at least",
    "189410 bytes (185kb) large.",
    "Unfortunately there were no such",
    "files found in the root directory.",
    "",
    "Please create some files and come",
    "back again.",
    NULL
}
```

Definition at line 73 of file [screen_photomode_save.c](#).

7.29.5.8 `int num_files_ok` `[static]`

Definition at line 35 of file [screen_photomode_save.c](#).

7.29.5.9 `const char* picked_file` `[static]`

Definition at line 39 of file [screen_photomode_save.c](#).

7.29.5.10 `SCREEN_STRUCT screen` `[static]`

Initial value:

```
= {
    enter,
    leave,
    update
}
```

Definition at line 332 of file [screen_photomode_save.c](#).

7.29.5.11 `enum { ... } state` `[static]`

7.30 [screen_photomode_save.c](#)

```
00001 /*****
00002 * Project:      discoverpixy
00003 * Website:     https://github.com/t-moe/discoverpixy
00004 * Authors:     Aaron Schmocker, Timo Lang
00005 * Institution:  BFH Bern University of Applied Sciences
00006 * File:        common/app/screen_photomode_save.c
00007 *
00008 * Version History:
00009 * Date          Autor Email          SHA          Changes
00010 * 2015-05-16    timolang@gmail.com    62006e0 Documented pixy_helper and implemented/finished photo-mode
00011 * 2015-06-07    timolang@gmail.com    c87220d Renamed pixy_helper to pixy_frame. Updated docu of appliaction.
00012 *              added doxygen comments to pixy_{frame,control}.h
00013 *****/
00014
00015 #include "screen_photomode_save.h"
00016 #include "filesystem.h"
00017 #include "button.h"
00018 #include "tft.h"
00019 #include "touch.h"
00020 #include "pixy.h"
00021 #include "pixy_frame.h"
00022 #include <stdlib.h>
00023 #include <string.h>
00024
00025
00026 static BUTTON_STRUCT b_back; //Button to navigate back
00027 static TOUCH_AREA_STRUCT a_area; //Touch area to select the save-file
00028
00029 //Callback for when the user presses the "back" button
00030 static void b_back_cb(void* button)
00031 {
00032     gui_screen_back();
00033 }
00034
00035 static int num_files_ok; //number of files into which we can write the image (size, flags ok)
00036 static enum {init, error, showlist, picking, saving,
00037             done} state; //Current state of the screen state machine
00038 static int fontheight; //The space between one line of text to the next
00039 static int liststart; //The y-Coordinate of the Start of the File-List
00040 static const char* picked_file; //The filename picked by the user, to save the image to
00041
00042 //Linked list structure to save all files which are suitable for saving.
00043 typedef struct FILE_LIST_ENTRY_S {
00044     char* filename; //Name of the file
00045     struct FILE_LIST_ENTRY_S* next; //Pointer to the next entry in the list or NULL
```



```

00045 } FILE_LIST_ENTRY;
00046
00047 static FILE_LIST_ENTRY* files_ok; //Pointer to the head of the list
00048
00049 //Callback for when the user selects a file to save the image into
00050 static void touchCB(void* touchArea, TOUCH_ACTION triggeredAction)
00051 {
00052     int y = touch_get_last_point().y - liststart; //Calculate the
00053     y-Coordinate of the touch point relative to the start of the file-list
00054     int elem = y / fontheight; //Calculate the file index
00055
00056     if (elem < 0 | elem >= num_files_ok) {
00057         return; //Check if the file index is valid (0,1,..,num_files_ok-1)
00058     }
00059
00060     //Search for the corresponding entry in the linked list
00061     FILE_LIST_ENTRY* current_entry = files_ok; //Start walking through the list,
    starting by the head of the list
00062
00063     for (int i = 0; i < elem; i++) { //Until we have reached the file (index)
00064         current_entry = current_entry->next; //traverse to the next file
00065     }
00066
00067     picked_file = current_entry->filename; //save the picked filename. It will be used
    by the statemachine in the main loop
00068     touch_unregister_area(&a_area); //unregister the touch area, we no longer need it.
    No more interrupts will be fired.
00069     state = saving; //Change the state of the statemachine
00070 }
00071
00072 //Text-Lines to show if we have no matching files (num_files_ok==0)
00073 static const char* nomatch_text [] = {
00074     "Due to limitations of the filesystem",
00075     "implementation you can only write to",
00076     "existing files.",
00077     "",
00078     "The files need to have a .bmp",
00079     "extension and must be at least",
00080     "189410 bytes (185kb) large.",
00081     "Unfortunately there were no such",
00082     "files found in the root directory.",
00083     "",
00084     "Please create some files and come",
00085     "back again.",
00086     NULL
00087 };
00088
00089
00090 //Bitmap header for a 318x198x24bit windows bitmap. data starts at 0x7A (= after this header)
00091 //This header has been taken from a white bitmap saved with gimp.
00092 //Wikipedia has a pretty good description on the header: http://de.wikipedia.org/wiki/Windows_Bitmap
00093 static unsigned char bmpheader_data[0x7A] = {
00094     0x42, 0x4d, 0xe2, 0xe3, 0x02, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x7a, 0x00,
00095     0x00, 0x00, 0x6c, 0x00, 0x00, 0x00, 0x3e, 0x01, 0x00, 0x00, 0xc6, 0x00,
00096     0x00, 0x00, 0x01, 0x00, 0x18, 0x00, 0x00, 0x00, 0x00, 0x00, 0x68, 0xe3,
00097     0x02, 0x00, 0x13, 0x0b, 0x00, 0x00, 0x13, 0x0b, 0x00, 0x00, 0x00, 0x00,
00098     0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x42, 0x47, 0x52, 0x73, 0x00, 0x00,
00099     0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
00100     0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
00101     0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
00102     0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x02, 0x00,
00103     0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
00104     0x00, 0x00
00105 };
00106
00107 //Callback for when the screen is entered/loaded
00108 static void enter(void* screen)
00109 {
00110     tft_clear(WHITE);
00111
00112     #define X_OFS 5
00113
00114     //Back button
00115     b_back.base.x1 = X_OFS; //Start X of Button
00116     b_back.base.y1 = 210; //Start Y of Button
00117     b_back.base.x2 = AUTO; //Auto Calculate X2 with String Width
00118     b_back.base.y2 = AUTO; //Auto Calculate Y2 with String Height
00119     b_back.txtcolor = WHITE; //Set foreground color
00120     b_back.bgcolor = HEX(0xAE1010); //Set background color (Don't take 255 or 0 on at least one
    channel, to make shadows possible)
00121     b_back.font = 0; //Select Font
00122     b_back.text = "Back"; //Set Text (For formatted strings take sprintf)
00123     b_back.callback = b_back_cb; //Call b_back_cb as Callback
00124     gui_button_add(&b_back); //Register Button (and run the callback from now on)
00125
00126

```

```

00127     state = init; //Start with the init state
00128     fontheight = tft_font_height(0) + 2; //Save the height of the used font, for
fast access
00129     files_ok = NULL; //initialize the linked list with 0 elements
00130     num_files_ok = 0; //we have zero! elements
00131 }
00132
00133 //Callback for when the screen should be updated
00134 //This is the main loop of the screen. This method will be called repeatedly
00135 static void update(void* screen)
00136 {
00137     switch (state) {
00138     case init: { //Init State: The user just entered the screen
00139         DIRECTORY_STRUCT* dir = filesystem_dir_open("."); //open root
directory
00140
00141         if (dir == NULL) { //error while opening root directory
00142             tft_print_line(X_OFS, 5, BLACK, TRANSPARENT, 0, "Error
accessing Filesystem");
00143             state = error;
00144             break;
00145         }
00146
00147         bool nomatch = true; //whether or not we have zero files which are suitable for saving
00148
00149         for (int i = 0; i < dir->num_files; i++) { //walk through all files in the directory
00150             FILE_STRUCT* file = &(dir->files[i]); //Pointer to the current
file/subdirectory
00151
00152             //Ignore directories, archives, hidden files, system files and files we cannot write to
00153             if (file->fattrib & (F_SYS | F_HID | F_ARC |
F_DIR | F_RDO)) {
00154                 continue;
00155             }
00156
00157             //ignore files which are not large enough
00158             if (file->fsize < 189410) {
00159                 continue; //size taken from an example bitmap (318x198x24)
00160             }
00161
00162             nomatch = false; //at least one file matches
00163             break;
00164         }
00165
00166         if (nomatch) { //not one file is suitable for writing
00167             int y = 5; //y-Coordinate where to start writing the error text
00168             int i = 0;
00169
00170             while (nomatch_text[i] != NULL) { //for every line in the big error array
00171                 //Write the line's text and go to the next line
00172                 tft_print_line(X_OFS, y + i * fontheight,
BLACK, TRANSPARENT, 0, nomatch_text[i]);
00173                 i++;
00174             }
00175
00176             state = error;
00177         } else { //we have a least one suitable file
00178             state = showlist;
00179         }
00180
00181         filesystem_dir_close(dir); //free directory struct
00182     }
00183     break;
00184
00185     case showlist: { //Show List State: Where we load and present the suitable file's to the user
in a list
00186         DIRECTORY_STRUCT* dir2 = filesystem_dir_open("."); //Open the
directory again
00187
00188         if (dir2 == NULL) {
00189             return; //Error on opening? This should never happen, since it's handled in the previous
state
00190         }
00191
00192         int y = 5; //y-Coordinate where to start drawing/writing text/list-elements
00193
00194         tft_print_line(X_OFS, y, BLACK, TRANSPARENT, 0, "Pick a file to
save the image to");
00195         y += fontheight + 5;
00196
00197         tft_print_line(X_OFS, y, BLUE, TRANSPARENT, 0, "Name
Modified      Size");
00198         y += fontheight;
00199
00200         liststart = y; //store the y coordinate of the start of the list away (used in toucharea
callback)
00201         num_files_ok = 0; //we start with 0 matching files

```

```

00202
00203     FILE_LIST_ENTRY* current_entry = NULL; //We start with an empty list
00204
00205     for (int i = 0; i < dir2->num_files && num_files_ok < 10; i++) { //go through
all the files of the directory, abort if we have 10 matches
00206         FILE_STRUCT* file = &(dir2->files[i]);
00207
00208         //Ignore directories, archives, hidden files, system files and files we cannot write to
00209         if (file->fattrib & (F_SYS | F_HID | F_ARC |
F_DIR | F_RDO)) {
00210             continue;
00211         }
00212
00213         //ignore files which are not large enough
00214         if (file->fsize < 189410) {
00215             continue; //size taken from an example bitmap (318x198x24)
00216         }
00217
00218         //Print out filename, modified date,time and file size
00219         tft_print_formatted(X_OFS, y, BLACK,
00220             TRANSPARENT, 0, "%-16s %02u.%02u.%02u %02u:%02u:%02u %u",
00221             file->fname,
00222             file->fdate.day,
00223             file->fdate.month,
00224             (file->fdate.year + 1980) % 100,
00225             file->ftime.hour,
00226             file->ftime.min,
00227             file->ftime.sec * 2,
00228             file->fsize);
00229
00230         if (current_entry == NULL) { //The list is empty
00231             current_entry = malloc(sizeof(FILE_LIST_ENTRY)); //create new entry
00232             files_ok = current_entry; //assign it to the list head
00233         } else { //there's a least one entry in the list
00234             current_entry->next = malloc(sizeof(FILE_LIST_ENTRY)); //append entry to
previous entry
00235             current_entry = current_entry->next; //newly created entry is the current now.
00236         }
00237
00238         current_entry->next = NULL; //we're at the end of the list (for now)
00239         current_entry->filename = malloc(strlen(file->fname) + 1); //allocate space for
the filename + zero-termination
00240         strcpy(current_entry->filename, file->fname); //copy filename (so that we can
close the directory after scanning)
00241
00242         //since we have found a suitable file we need to increment the position in the list
00243         num_files_ok++;
00244         y += fontheight;
00245     }
00246
00247     //Touch area for file-selection (in the list)
00248     a_area.hookedActions = PEN_UP; //we're only interested in PEN_UP events
00249     a_area.x1 = X_OFS; //Left border
00250     a_area.y1 = liststart; //Start where the list started
00251     a_area.x2 = 320 - X_OFS; //Right border
00252     a_area.y2 = liststart + fontheight * num_files_ok; //stop at the
end of the list
00253     a_area.callback = touchCB; //execute our callback when PEN_UP occurs
00254     touch_register_area(&a_area); //register the touch area and receive events from
now on
00255
00256     filesystem_dir_close(dir2); //we no longer need the directory struct, since we
have our own linked list now
00257
00258     state = picking;
00259 }
00260 break;
00261
00262 case picking: //Picking State: Where we wait on the users file choice
00263     pixy_service(); //Handle pending pixy events
00264     //do nothing and wait on user to pick a file
00265     break;
00266
00267 case saving: { //Saving State: Where we save the image to the selected file
00268     FILE_HANDLE* file = filesystem_file_open(
picked_file); //try to open the selected file
00269
00270     if (file == NULL) { //opening the file failed
00271         tft_print_formatted(X_OFS, 190, BLUE,
TRANSPARENT, 0, "Could not open %s", picked_file);
00272         state = error;
00273         break;
00274     }
00275
00276     filesystem_file_seek(file, 0); //seek to the start of the file (optional?)
00277
00278     if (filesystem_file_write(file, bmpheader_data, 0x7A) !=

```

```

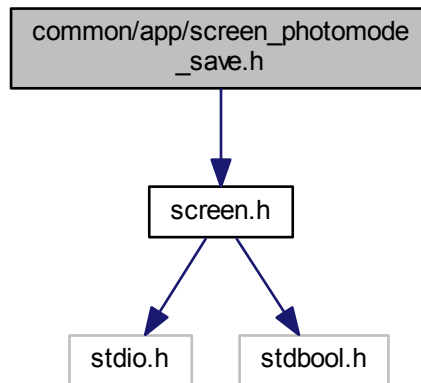
F_OK) { //Writing the header failed
00279     tft_print_formatted(X_OFS, 190, BLUE,
TRANSPARENT, 0, "Error while writing to %s", picked_file);
00280     filesystem_file_close(file);
00281     state = error;
00282     break;
00283 }
00284
00285     if (pixy_save_full_frame(file) != 0) { //Writing the imagedata failed
00286     tft_print_formatted(X_OFS, 190, BLUE,
TRANSPARENT, 0, "Error while writing to %s", picked_file);
00287     filesystem_file_close(file);
00288     state = error;
00289     break;
00290 }
00291
00292     //if we reach this point, we have written all data out successfully
00293
00294     filesystem_file_close(file); //close/finalize the file
00295     tft_print_formatted(X_OFS, 190, BLUE,
TRANSPARENT, 0, "Image saved to %s", picked_file);
00296     state = done;
00297 }
00298 break;
00299
00300     case error: //Error State: Where we show an error message and leave the user no other choice than
to click the backbutton
00301     case done: //Done State: When saving the file was successful
00302     pixy_service(); //Handle pending pixy events
00303     //wait on user to click the back button
00304     break;
00305 }
00306 }
00307 }
00308
00309 //Callback for when the screen is left/unloaded
00310 static void leave(void* screen)
00311 {
00312     gui_button_remove(&b_back); //Remove/Free the back button
00313
00314     if (state == picking) { //The user left the screen in the "picking"-phase
00315         touch_unregister_area(&a_area); //remove the touch area (for the list)
00316     }
00317
00318     if (state == picking || state == saving || state ==
done) { //the user left the screen after we created the linked list
00319         //Iterate through the linked list and free all resources
00320         FILE_LIST_ENTRY* current_entry = files_ok; //start with the list head
00321
00322         while (current_entry != NULL) { //while we're not at the end
00323             FILE_LIST_ENTRY* temp = current_entry->next; //save the next pointer because
we free the current element on the next line
00324             free((void*)(current_entry->filename)); //free filename
00325             free(current_entry); //free element itself
00326             current_entry = temp; //advance
00327         }
00328     }
00329 }
00330
00331 //Declare screen callbacks
00332 static SCREEN_STRUCT screen = {
00333     enter,
00334     leave,
00335     update
00336 };
00337
00338 SCREEN_STRUCT* get_screen_photomodesave()
00339 {
00340     return &screen;
00341 }

```

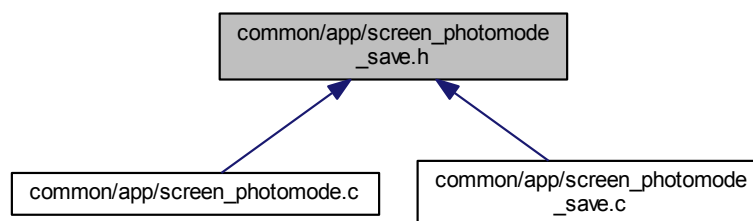
7.31 common/app/screen_photomode_save.h File Reference

```
#include "screen.h"
```

Include dependency graph for screen_photomode_save.h:



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



Functions

- `SCREEN_STRUCT * get_screen_photomodesave ()`

7.32 screen_photomode_save.h

```

00001 /*****
00002  * Project:      discoverpixy
00003  * Website:     https://github.com/t-moe/discoverpixy
00004  * Authors:     Aaron Schmocker, Timo Lang
00005  * Institution:  BFH Bern University of Applied Sciences
00006  * File:        common/app/screen_photomode_save.h
00007  *
00008  * Version History:
00009  * Date          Autor Email          SHA          Changes
00010  * 2015-05-16    timolang@gmail.com    62006e0    Documented pixy_helper and implemented/finished photo-mode
  
```

```

        screens! Snap some shots!
00011 *
00012 *****/
00013
00014 #include "screen.h"
00015
00020
00026
00032 SCREEN_STRUCT* get_screen_photomodesave();
00033

```

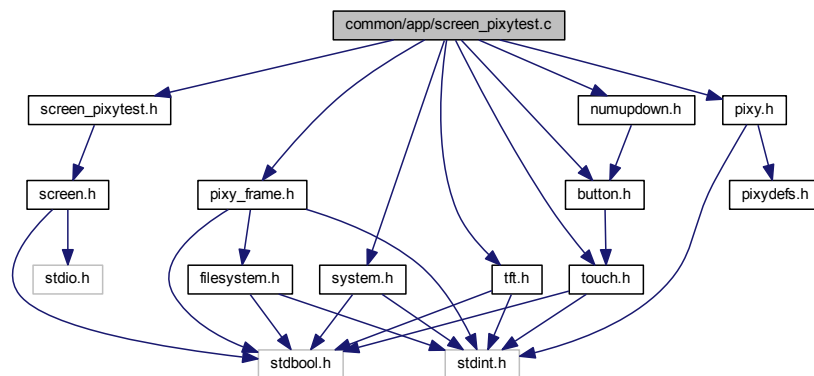
7.33 common/app/screen_pixytest.c File Reference

```

#include "screen_pixytest.h"
#include "button.h"
#include "numupdown.h"
#include "tft.h"
#include "touch.h"
#include "pixy.h"
#include "system.h"
#include "pixy_frame.h"

```

Include dependency graph for screen_pixytest.c:



Macros

- #define [SERVO_BUTTON_Y](#) 10
- #define [SERVO_BUTTON_SPACING](#) 5
- #define [LED_COLOR_BUTTON_Y](#) 35
- #define [LED_COLOR_BUTTON_SPACING](#) 5
- #define [LED_POWER_BUTTON_Y](#) 70

Enumerations

- enum {
[detecting](#), [idle](#), [update_servos](#), [update_ledcolor](#),
[update_ledcurrent](#) }

Functions

- static void [b_back_cb](#) (void *button)

- static void [b_servos_center_cb](#) (void *button)
- static void [b_servos_topleft_cb](#) (void *button)
- static void [b_servos_topright_cb](#) (void *button)
- static void [b_servos_bottomleft_cb](#) (void *button)
- static void [b_servos_bottomright_cb](#) (void *button)
- static void [b_led_off_cb](#) (void *button)
- static void [b_led_white_cb](#) (void *button)
- static void [b_led_red_cb](#) (void *button)
- static void [b_led_green_cb](#) (void *button)
- static void [b_led_blue_cb](#) (void *button)
- static void [n_led_powerlimit_cb](#) (void *numupdown, int16_t value)
- static void [enter](#) (void *screen)
- static void [leave](#) (void *screen)
- static void [update](#) (void *screen)
- [SCREEN_STRUCT](#) * [get_screen_pixytest](#) ()

Variables

- static enum { ... } [state](#)
- static [BUTTON_STRUCT](#) [b_back](#)
- static [BUTTON_STRUCT](#) [b_servos_center](#)
- static [BUTTON_STRUCT](#) [b_servos_topleft](#)
- static [BUTTON_STRUCT](#) [b_servos_topright](#)
- static [BUTTON_STRUCT](#) [b_servos_bottomleft](#)
- static [BUTTON_STRUCT](#) [b_servos_bottomright](#)
- static uint16_t [servo_x](#)
- static uint16_t [servo_y](#)
- static [BUTTON_STRUCT](#) [b_led_off](#)
- static [BUTTON_STRUCT](#) [b_led_white](#)
- static [BUTTON_STRUCT](#) [b_led_red](#)
- static [BUTTON_STRUCT](#) [b_led_green](#)
- static [BUTTON_STRUCT](#) [b_led_blue](#)
- static uint32_t [led_color](#)
- static uint32_t [led_maxcurrent](#)
- static [NUMUPDOWN_STRUCT](#) [n_led_powerlimit](#)
- static [SCREEN_STRUCT](#) [screen](#)

7.33.1 Macro Definition Documentation

7.33.1.1 `#define LED_COLOR_BUTTON_SPACING 5`

7.33.1.2 `#define LED_COLOR_BUTTON_Y 35`

7.33.1.3 `#define LED_POWER_BUTTON_Y 70`

7.33.1.4 `#define SERVO_BUTTON_SPACING 5`

7.33.1.5 `#define SERVO_BUTTON_Y 10`

7.33.2 Enumeration Type Documentation

7.33.2.1 anonymous enum

Enumerator

detecting

*idle**update_servos**update_ledcolor**update_ledcurrent*

Definition at line 30 of file [screen_pixytest.c](#).

```
00030 {detecting, idle, update_servos, update_ledcolor,
      update_ledcurrent} state; //Current state of the screen state machine
```

7.33.3 Function Documentation

7.33.3.1 static void b_back_cb (void * *button*) [static]

Definition at line 53 of file [screen_pixytest.c](#).

```
00054 {
00055     gui_screen_back();
00056 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



7.33.3.2 static void b_led_blue_cb (void * *button*) [static]

Definition at line 135 of file [screen_pixytest.c](#).

```
00136 {
00137     if (state == idle) {
00138         led_color = 0x0000FF;
00139         state = update_ledcolor;
00140     }
00141 }
```


Here is the caller graph for this function:

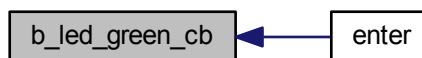


7.33.3.3 static void b_led_green_cb (void * *button*) [static]

Definition at line 127 of file [screen_pixytest.c](#).

```
00128 {  
00129     if (state == idle) {  
00130         led_color = 0x00FF00;  
00131         state = update_ledcolor;  
00132     }  
00133 }
```

Here is the caller graph for this function:



7.33.3.4 static void b_led_off_cb (void * *button*) [static]

Definition at line 103 of file [screen_pixytest.c](#).

```
00104 {  
00105     if (state == idle) {  
00106         led_color = 0x000000;  
00107         state = update_ledcolor;  
00108     }  
00109 }
```

Here is the caller graph for this function:

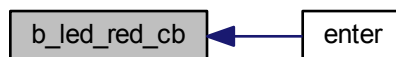


7.33.3.5 static void b_led_red_cb (void * *button*) [static]

Definition at line 119 of file [screen_pixytest.c](#).

```
00120 {  
00121     if (state == idle) {  
00122         led_color = 0xFF0000;  
00123         state = update_ledcolor;  
00124     }  
00125 }
```

Here is the caller graph for this function:

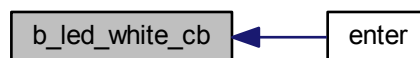


7.33.3.6 static void b_led_white_cb (void * *button*) [static]

Definition at line 111 of file [screen_pixytest.c](#).

```
00112 {  
00113     if (state == idle) {  
00114         led_color = 0xFFFFFF;  
00115         state = update_ledcolor;  
00116     }  
00117 }
```

Here is the caller graph for this function:

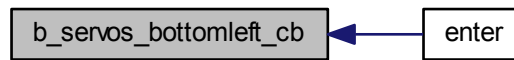


7.33.3.7 static void b_servos_bottomleft_cb (void * *button*) [static]

Definition at line 85 of file [screen_pixytest.c](#).

```
00086 {  
00087     if (state == idle) {  
00088         servo_x = 0;  
00089         servo_y = 1000;  
00090         state = update_servos;  
00091     }  
00092 }
```

Here is the caller graph for this function:

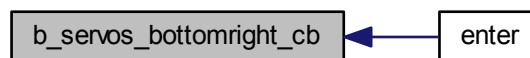


7.33.3.8 static void b_servos_bottomright_cb (void * *button*) [static]

Definition at line 94 of file [screen_pixytest.c](#).

```
00095 {  
00096     if (state == idle) {  
00097         servo_x = 1000;  
00098         servo_y = 1000;  
00099         state = update_servos;  
00100     }  
00101 }
```

Here is the caller graph for this function:

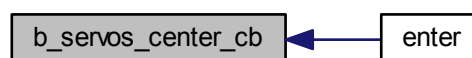


7.33.3.9 static void b_servos_center_cb (void * *button*) [static]

Definition at line 58 of file [screen_pixytest.c](#).

```
00059 {  
00060     if (state == idle) {  
00061         servo_x = 500;  
00062         servo_y = 500;  
00063         state = update_servos;  
00064     }  
00065 }
```

Here is the caller graph for this function:

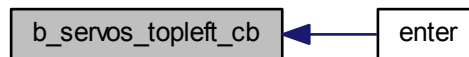


7.33.3.10 static void b_servos_topleft_cb (void * *button*) [static]

Definition at line 67 of file [screen_pixytest.c](#).

```
00068 {
00069     if (state == idle) {
00070         servo_x = 0;
00071         servo_y = 0;
00072         state = update_servos;
00073     }
00074 }
```

Here is the caller graph for this function:

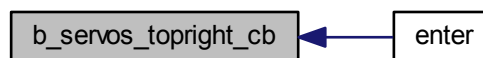


7.33.3.11 static void b_servos_topright_cb (void * *button*) [static]

Definition at line 76 of file [screen_pixytest.c](#).

```
00077 {
00078     if (state == idle) {
00079         servo_x = 1000;
00080         servo_y = 0;
00081         state = update_servos;
00082     }
00083 }
```

Here is the caller graph for this function:



7.33.3.12 static void enter (void * *screen*) [static]

Definition at line 151 of file [screen_pixytest.c](#).

```
00152 {
00153     tft_clear(WHITE);
00154
00155     //Back button
00156     b_back.base.x1 = 10; //Start X of Button
00157     b_back.base.y1 = 210; //Start Y of Button
00158     b_back.base.x2 = AUTO; //Auto Calculate X2 with String Width
00159     b_back.base.y2 = AUTO; //Auto Calculate Y2 with String Height
  
```

```

00160     b_back.txtcolor = WHITE; //Set foreground color
00161     b_back.bgcolor = HEX(0xAE1010); //Set background color (Don't take 255 or 0 on at least
one channel, to make shadows possible)
00162     b_back.font = 0; //Select Font
00163     b_back.text = "Back"; //Set Text (For formatted strings take sprintf)
00164     b_back.callback = b_back_cb; //Call b_back_cb as Callback
00165     gui_button_add(&b_back); //Register Button (and run the callback from now on)
00166
00167
00168     //Servo stuff
00169     #define SERVO_BUTTON_Y 10
00170     #define SERVO_BUTTON_SPACING 5
00171     tft_print_line(5, SERVO_BUTTON_Y, BLACK,
TRANSPARENT, 0, "Servos:");
00172
00173     b_servos_center.base.x1 = 55;
00174     b_servos_center.base.y1 = SERVO_BUTTON_Y - 3;
00175     b_servos_center.base.x2 = AUTO;
00176     b_servos_center.base.y2 = AUTO;
00177     b_servos_center.txtcolor = WHITE;
00178     b_servos_center.bgcolor = HEX(0xAE1010);
00179     b_servos_center.font = 0;
00180     b_servos_center.text = "Center";
00181     b_servos_center.callback = b_servos_center_cb;
00182     gui_button_add(&b_servos_center);
00183
00184     b_servos_topleft.base.x1 = b_servos_center.
base.x2 + SERVO_BUTTON_SPACING;
00185     b_servos_topleft.base.y1 = SERVO_BUTTON_Y - 3;
00186     b_servos_topleft.base.x2 = AUTO;
00187     b_servos_topleft.base.y2 = AUTO;
00188     b_servos_topleft.txtcolor = WHITE;
00189     b_servos_topleft.bgcolor = HEX(0xAE1010);
00190     b_servos_topleft.font = 0;
00191     b_servos_topleft.text = "ToLe";
00192     b_servos_topleft.callback = b_servos_topleft_cb;
00193     gui_button_add(&b_servos_topleft);
00194
00195     b_servos_topright.base.x1 = b_servos_topleft.
base.x2 + SERVO_BUTTON_SPACING;
00196     b_servos_topright.base.y1 = SERVO_BUTTON_Y - 3;
00197     b_servos_topright.base.x2 = AUTO;
00198     b_servos_topright.base.y2 = AUTO;
00199     b_servos_topright.txtcolor = WHITE;
00200     b_servos_topright.bgcolor = HEX(0xAE1010);
00201     b_servos_topright.font = 0;
00202     b_servos_topright.text = "ToRi";
00203     b_servos_topright.callback = b_servos_topright_cb;
00204     gui_button_add(&b_servos_topright);
00205
00206     b_servos_bottomleft.base.x1 = b_servos_topright.
base.x2 + SERVO_BUTTON_SPACING;
00207     b_servos_bottomleft.base.y1 = SERVO_BUTTON_Y - 3;
00208     b_servos_bottomleft.base.x2 = AUTO;
00209     b_servos_bottomleft.base.y2 = AUTO;
00210     b_servos_bottomleft.txtcolor = WHITE;
00211     b_servos_bottomleft.bgcolor = HEX(0xAE1010);
00212     b_servos_bottomleft.font = 0;
00213     b_servos_bottomleft.text = "BoLe";
00214     b_servos_bottomleft.callback =
b_servos_bottomleft_cb;
00215     gui_button_add(&b_servos_bottomleft);
00216
00217     b_servos_bottomright.base.x1 = b_servos_bottomleft.
base.x2 + SERVO_BUTTON_SPACING;
00218     b_servos_bottomright.base.y1 = SERVO_BUTTON_Y - 3;
00219     b_servos_bottomright.base.x2 = AUTO;
00220     b_servos_bottomright.base.y2 = AUTO;
00221     b_servos_bottomright.txtcolor = WHITE;
00222     b_servos_bottomright.bgcolor = HEX(0xAE1010);
00223     b_servos_bottomright.font = 0;
00224     b_servos_bottomright.text = "BoRi";
00225     b_servos_bottomright.callback =
b_servos_bottomright_cb;
00226     gui_button_add(&b_servos_bottomright);
00227
00228     //Led Color stuff
00229     #define LED_COLOR_BUTTON_Y 35
00230     #define LED_COLOR_BUTTON_SPACING 5
00231     tft_print_line(5, LED_COLOR_BUTTON_Y, BLACK,
TRANSPARENT, 0, "Led Color:");
00232
00233     b_led_off.base.x1 = 85;
00234     b_led_off.base.y1 = LED_COLOR_BUTTON_Y - 3;
00235     b_led_off.base.x2 = AUTO;
00236     b_led_off.base.y2 = AUTO;
00237     b_led_off.txtcolor = WHITE;

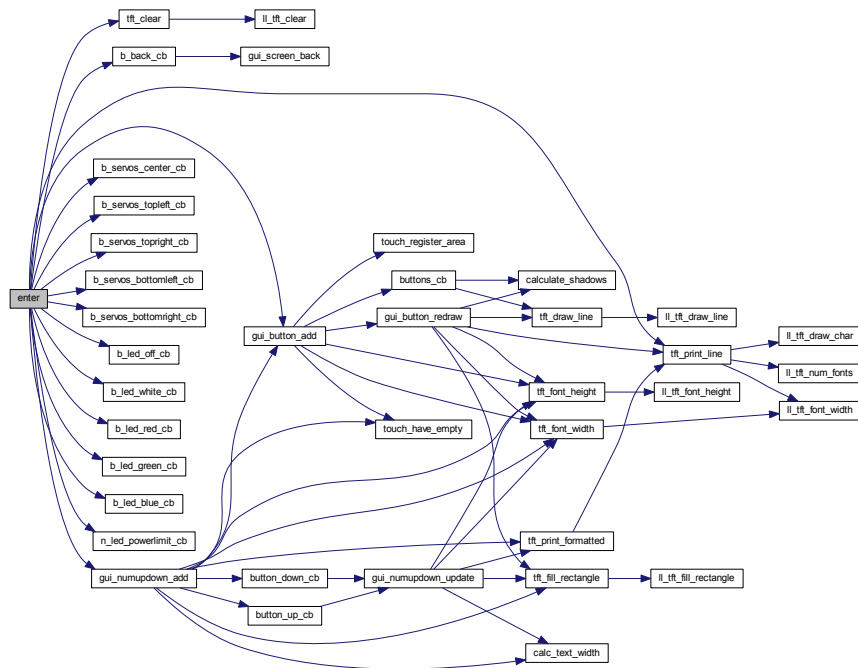
```

```

00238     b_led_off.bgcolor = BLACK;
00239     b_led_off.font = 0;
00240     b_led_off.text = "Off";
00241     b_led_off.callback = b_led_off_cb;
00242     gui_button_add(&b_led_off);
00243
00244     b_led_white.base.x1 = b_led_off.base.x2 +
LED_COLOR_BUTTON_SPACING;
00245     b_led_white.base.y1 = LED_COLOR_BUTTON_Y - 3;
00246     b_led_white.base.x2 = AUTO;
00247     b_led_white.base.y2 = AUTO;
00248     b_led_white.txtcolor = BLACK;
00249     b_led_white.bgcolor = HEX(0xEEEEEE);
00250     b_led_white.font = 0;
00251     b_led_white.text = "White";
00252     b_led_white.callback = b_led_white_cb;
00253     gui_button_add(&b_led_white);
00254
00255     b_led_red.base.x1 = b_led_white.base.x2 +
LED_COLOR_BUTTON_SPACING;
00256     b_led_red.base.y1 = LED_COLOR_BUTTON_Y - 3;
00257     b_led_red.base.x2 = AUTO;
00258     b_led_red.base.y2 = AUTO;
00259     b_led_red.txtcolor = WHITE;
00260     b_led_red.bgcolor = HEX(0xEE0000);
00261     b_led_red.font = 0;
00262     b_led_red.text = "Red";
00263     b_led_red.callback = b_led_red_cb;
00264     gui_button_add(&b_led_red);
00265
00266     b_led_green.base.x1 = b_led_red.base.x2 +
LED_COLOR_BUTTON_SPACING;
00267     b_led_green.base.y1 = LED_COLOR_BUTTON_Y - 3;
00268     b_led_green.base.x2 = AUTO;
00269     b_led_green.base.y2 = AUTO;
00270     b_led_green.txtcolor = WHITE;
00271     b_led_green.bgcolor = HEX(0x00EE00);
00272     b_led_green.font = 0;
00273     b_led_green.text = "Green";
00274     b_led_green.callback = b_led_green_cb;
00275     gui_button_add(&b_led_green);
00276
00277     b_led_blue.base.x1 = b_led_green.base.x2 +
LED_COLOR_BUTTON_SPACING;
00278     b_led_blue.base.y1 = LED_COLOR_BUTTON_Y - 3;
00279     b_led_blue.base.x2 = AUTO;
00280     b_led_blue.base.y2 = AUTO;
00281     b_led_blue.txtcolor = WHITE;
00282     b_led_blue.bgcolor = HEX(0x0000EE);
00283     b_led_blue.font = 0;
00284     b_led_blue.text = "Blue";
00285     b_led_blue.callback = b_led_blue_cb;
00286     gui_button_add(&b_led_blue);
00287
00288     //Led MaxPower stuff
00289     #define LED_POWER_BUTTON_Y 70
00290     tft_print_line(5, LED_POWER_BUTTON_Y, BLACK,
TRANSPARENT, 0, "Led Maximum Current:");
00291
00292     //Num up down test
00293     n_led_powerlimit.x = 160;
00294     n_led_powerlimit.y = LED_POWER_BUTTON_Y - 7;
00295     n_led_powerlimit.fgcolor = WHITE;
00296     n_led_powerlimit.value = 10;
00297     n_led_powerlimit.max = 40;
00298     n_led_powerlimit.min = 0;
00299     n_led_powerlimit.callback = n_led_powerlimit_cb;
00300     gui_numupdown_add(&n_led_powerlimit);
00301
00302
00303
00304     state = detecting;
00305 }

```

Here is the call graph for this function:

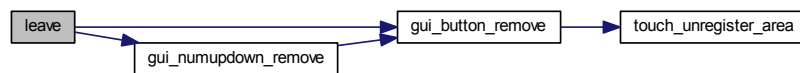


7.33.3.13 static void leave (void * screen) [static]

Definition at line 307 of file [screen_pixytest.c](#).

```
00308 {
00309     gui_button_remove (&b_back);
00310     gui_button_remove (&b_servos_center);
00311     gui_button_remove (&b_servos_topleft);
00312     gui_button_remove (&b_servos_topright);
00313     gui_button_remove (&b_servos_bottomleft);
00314     gui_button_remove (&b_servos_bottomright);
00315     gui_button_remove (&b_led_off);
00316     gui_button_remove (&b_led_white);
00317     gui_button_remove (&b_led_red);
00318     gui_button_remove (&b_led_green);
00319     gui_button_remove (&b_led_blue);
00320     gui_numupdown_remove (&n_led_powerlimit);
00321 }
00322 }
```

Here is the call graph for this function:



7.33.3.14 static void n_led_powerlimit_cb (void * numupdown, int16_t value) [static]

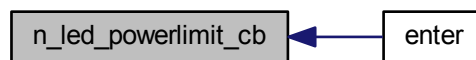
Definition at line 143 of file [screen_pixytest.c](#).

```

00144 {
00145     if (state == idle) {
00146         led_maxcurrent = value;
00147         state = update_ledcurrent;
00148     }
00149 }

```

Here is the caller graph for this function:



7.33.3.15 static void update (void * screen) [static]

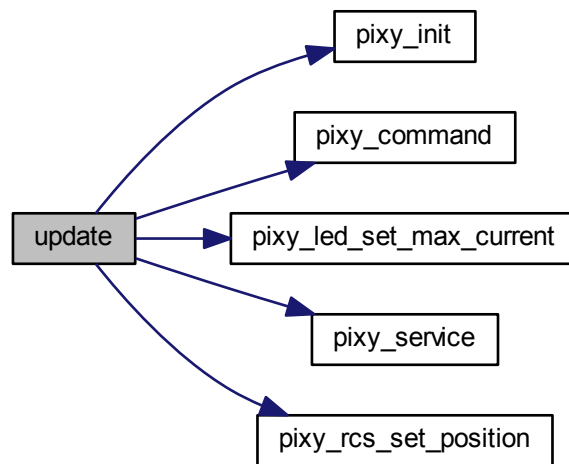
Definition at line 325 of file [screen_pixytest.c](#).

```

00326 {
00327     switch (state) {
00328     case detecting: //Detecting State: Where we try to connect to the pixy
00329         if (pixy_init() == 0) { //Pixy connection ok
00330             int32_t response;
00331             int return_value;
00332             return_value = pixy_command("stop", END_OUT_ARGS, &response,
END_IN_ARGS);
00333             pixy_led_set_max_current(10);
00334             state = idle; //Go to next state
00335         }
00336         break;
00337     case idle:
00338         pixy_service();
00339         break;
00340     case update_servos:
00341         pixy_rcs_set_position(0, servo_x);
00342         pixy_rcs_set_position(1, servo_y);
00343         state = idle;
00344         break;
00345     case update_ledcolor: {
00346         int32_t response;
00347         int return_value;
00348         return_value = pixy_command("led_set", INT32(led_color),
END_OUT_ARGS, &response, END_IN_ARGS);
00349         state = idle;
00350     }
00351     break;
00352     case update_ledcurrent:
00353         pixy_led_set_max_current(led_maxcurrent);
00354         state = idle;
00355         break;
00356     }
00357 }
00358 }
00359 }
00360 }
00361 }
00362 }
00363 }
00364 }

```


Here is the call graph for this function:



7.33.4 Variable Documentation

7.33.4.1 `BUTTON_STRUCT b_back` `[static]`

Definition at line 32 of file [screen_pixytest.c](#).

7.33.4.2 `BUTTON_STRUCT b_led_blue` `[static]`

Definition at line 46 of file [screen_pixytest.c](#).

7.33.4.3 `BUTTON_STRUCT b_led_green` `[static]`

Definition at line 45 of file [screen_pixytest.c](#).

7.33.4.4 `BUTTON_STRUCT b_led_off` `[static]`

Definition at line 42 of file [screen_pixytest.c](#).

7.33.4.5 `BUTTON_STRUCT b_led_red` `[static]`

Definition at line 44 of file [screen_pixytest.c](#).

7.33.4.6 `BUTTON_STRUCT b_led_white` `[static]`

Definition at line 43 of file [screen_pixytest.c](#).

7.33.4.7 **BUTTON_STRUCT** `b_servos_bottomleft` `[static]`

Definition at line 37 of file [screen_pixytest.c](#).

7.33.4.8 **BUTTON_STRUCT** `b_servos_bottomright` `[static]`

Definition at line 38 of file [screen_pixytest.c](#).

7.33.4.9 **BUTTON_STRUCT** `b_servos_center` `[static]`

Definition at line 34 of file [screen_pixytest.c](#).

7.33.4.10 **BUTTON_STRUCT** `b_servos_topleft` `[static]`

Definition at line 35 of file [screen_pixytest.c](#).

7.33.4.11 **BUTTON_STRUCT** `b_servos_topright` `[static]`

Definition at line 36 of file [screen_pixytest.c](#).

7.33.4.12 **uint32_t** `led_color` `[static]`

Definition at line 47 of file [screen_pixytest.c](#).

7.33.4.13 **uint32_t** `led_maxcurrent` `[static]`

Definition at line 49 of file [screen_pixytest.c](#).

7.33.4.14 **NUMUPDOWN_STRUCT** `n_led_powerlimit` `[static]`

Definition at line 50 of file [screen_pixytest.c](#).

7.33.4.15 **SCREEN_STRUCT** `screen` `[static]`

Initial value:

```
= {  
    enter,  
    leave,  
    update  
}
```

Definition at line 367 of file [screen_pixytest.c](#).

7.33.4.16 **uint16_t** `servo_x` `[static]`

Definition at line 39 of file [screen_pixytest.c](#).

7.33.4.17 **uint16_t** `servo_y` `[static]`

Definition at line 40 of file [screen_pixytest.c](#).

7.33.4.18 enum { ... } state [static]

7.34 screen_pixytest.c

```

00001 /*****
00002  * Project:      discoverpixy
00003  * Website:     https://github.com/t-moe/discoverpixy
00004  * Authors:     Aaron Schmocker, Timo Lang
00005  * Institution:  BFH Bern University of Applied Sciences
00006  * File:        common/app/screen_pixytest.c
00007  *
00008  * Version History:
00009  * Date      Autor Email      SHA      Changes
00010  * 2015-04-27 timolang@gmail.com cf72baa Introduced a Screen (sub) module and divided app into multiple
00011  * 2015-05-02 timolang@gmail.com 3281616 Added some more touch functions. Improved pixy test. Drag the
00012  * 2015-05-09 timolang@gmail.com 0b5173e Added reference tracking.
00013  * 2015-05-15 timolang@gmail.com 27c09ba Redesigned main menu. Moved stuff from pixytest to a new helper
00014  * 2015-05-25 timolang@gmail.com 6a61769 Reimplemented pixytest screen. Added a lot of Test-Buttons.
00015  * 2015-06-01 aaron@duckpond.ch caa7b5c Added IRQ for user button, fixed some touchproblems.
00016  * 2015-06-03 timolang@gmail.com 74aa186 Made pixy_test screen working again. Had to disable
00017  * 2015-06-07 timolang@gmail.com c87220d Renamed pixy_helper to pixy_frame. Updated docu of appliaction.
00018  * added doxygen comments to pixy_{frame,control}.h
00019  *****/
00020
00021 #include "screen_pixytest.h"
00022 #include "button.h"
00023 #include "numupdown.h"
00024 #include "tft.h"
00025 #include "touch.h"
00026 #include "pixy.h"
00027 #include "system.h"
00028 #include "pixy_frame.h"
00029
00030 static volatile enum {detecting, idle, update_servos,
00031                      update_ledcolor, update_ledcurrent} state; //Current state of the
00032                      screen state machine
00033
00034 static BUTTON_STRUCT b_back;
00035
00036 static BUTTON_STRUCT b_servos_center;
00037 static BUTTON_STRUCT b_servos_topleft;
00038 static BUTTON_STRUCT b_servos_topright;
00039 static BUTTON_STRUCT b_servos_bottomleft;
00040 static BUTTON_STRUCT b_servos_bottomright;
00041
00042 static uint16_t servo_x;
00043 static uint16_t servo_y;
00044
00045 static BUTTON_STRUCT b_led_off;
00046 static BUTTON_STRUCT b_led_white;
00047 static BUTTON_STRUCT b_led_red;
00048 static BUTTON_STRUCT b_led_green;
00049 static BUTTON_STRUCT b_led_blue;
00050
00051 static uint32_t led_color;
00052
00053 static uint32_t led_maxcurrent;
00054 static NUMUPDOWN_STRUCT n_led_powerlimit;
00055
00056 static void b_back_cb(void* button)
00057 {
00058     gui_screen_back();
00059 }
00060
00061 static void b_servos_center_cb(void* button)
00062 {
00063     if (state == idle) {
00064         servo_x = 500;
00065         servo_y = 500;
00066         state = update_servos;
00067     }
00068 }
00069
00070 static void b_servos_topleft_cb(void* button)
00071 {
00072     if (state == idle) {
00073         servo_x = 0;
00074         servo_y = 0;
00075     }
00076 }

```

```

00072         state = update_servos;
00073     }
00074 }
00075
00076 static void b_servos_topright_cb(void* button)
00077 {
00078     if (state == idle) {
00079         servo_x = 1000;
00080         servo_y = 0;
00081         state = update_servos;
00082     }
00083 }
00084
00085 static void b_servos_bottomleft_cb(void* button)
00086 {
00087     if (state == idle) {
00088         servo_x = 0;
00089         servo_y = 1000;
00090         state = update_servos;
00091     }
00092 }
00093
00094 static void b_servos_bottomright_cb(void* button)
00095 {
00096     if (state == idle) {
00097         servo_x = 1000;
00098         servo_y = 1000;
00099         state = update_servos;
00100     }
00101 }
00102
00103 static void b_led_off_cb(void* button)
00104 {
00105     if (state == idle) {
00106         led_color = 0x000000;
00107         state = update_ledcolor;
00108     }
00109 }
00110
00111 static void b_led_white_cb(void* button)
00112 {
00113     if (state == idle) {
00114         led_color = 0xFFFFFF;
00115         state = update_ledcolor;
00116     }
00117 }
00118
00119 static void b_led_red_cb(void* button)
00120 {
00121     if (state == idle) {
00122         led_color = 0xFF0000;
00123         state = update_ledcolor;
00124     }
00125 }
00126
00127 static void b_led_green_cb(void* button)
00128 {
00129     if (state == idle) {
00130         led_color = 0x00FF00;
00131         state = update_ledcolor;
00132     }
00133 }
00134
00135 static void b_led_blue_cb(void* button)
00136 {
00137     if (state == idle) {
00138         led_color = 0x0000FF;
00139         state = update_ledcolor;
00140     }
00141 }
00142
00143 static void n_led_powerlimit_cb(void* numupdown, int16_t value)
00144 {
00145     if (state == idle) {
00146         led_maxcurrent = value;
00147         state = update_ledcurrent;
00148     }
00149 }
00150
00151 static void enter(void* screen)
00152 {
00153     tft_clear(WHITE);
00154
00155     //Back button
00156     b_back.base.x1 = 10; //Start X of Button
00157     b_back.base.y1 = 210; //Start Y of Button
00158     b_back.base.x2 = AUTO; //Auto Calculate X2 with String Width

```

```

00159     b_back.base.y2 = AUTO; //Auto Calculate Y2 with String Height
00160     b_back.txtcolor = WHITE; //Set foreground color
00161     b_back.bgcolor = HEX(0xAE1010); //Set background color (Don't take 255 or 0 on at least one
channel, to make shadows possible)
00162     b_back.font = 0; //Select Font
00163     b_back.text = "Back"; //Set Text (For formatted strings take sprintf)
00164     b_back.callback = b_back_cb; //Call b_back_cb as Callback
00165     gui_button_add(&b_back); //Register Button (and run the callback from now on)
00166
00167     //Servo stuff
00168     #define SERVO_BUTTON_Y 10
00169     #define SERVO_BUTTON_SPACING 5
00170     tft_print_line(5, SERVO_BUTTON_Y, BLACK,
TRANSPARENT, 0, "Servos:");
00171
00172     b_servos_center.base.x1 = 55;
00173     b_servos_center.base.y1 = SERVO_BUTTON_Y - 3;
00174     b_servos_center.base.x2 = AUTO;
00175     b_servos_center.base.y2 = AUTO;
00176     b_servos_center.txtcolor = WHITE;
00177     b_servos_center.bgcolor = HEX(0xAE1010);
00178     b_servos_center.font = 0;
00179     b_servos_center.text = "Center";
00180     b_servos_center.callback = b_servos_center_cb;
00181     gui_button_add(&b_servos_center);
00182
00183     b_servos_topleft.base.x1 = b_servos_center.base.x2 +
SERVO_BUTTON_SPACING;
00184     b_servos_topleft.base.y1 = SERVO_BUTTON_Y - 3;
00185     b_servos_topleft.base.x2 = AUTO;
00186     b_servos_topleft.base.y2 = AUTO;
00187     b_servos_topleft.txtcolor = WHITE;
00188     b_servos_topleft.bgcolor = HEX(0xAE1010);
00189     b_servos_topleft.font = 0;
00190     b_servos_topleft.text = "ToLe";
00191     b_servos_topleft.callback = b_servos_topleft_cb;
00192     gui_button_add(&b_servos_topleft);
00193
00194     b_servos_topright.base.x1 = b_servos_topleft.base.x2 +
SERVO_BUTTON_SPACING;
00195     b_servos_topright.base.y1 = SERVO_BUTTON_Y - 3;
00196     b_servos_topright.base.x2 = AUTO;
00197     b_servos_topright.base.y2 = AUTO;
00198     b_servos_topright.txtcolor = WHITE;
00199     b_servos_topright.bgcolor = HEX(0xAE1010);
00200     b_servos_topright.font = 0;
00201     b_servos_topright.text = "ToRi";
00202     b_servos_topright.callback = b_servos_topright_cb;
00203     gui_button_add(&b_servos_topright);
00204
00205     b_servos_bottomleft.base.x1 = b_servos_topright.base.x2 +
SERVO_BUTTON_SPACING;
00206     b_servos_bottomleft.base.y1 = SERVO_BUTTON_Y - 3;
00207     b_servos_bottomleft.base.x2 = AUTO;
00208     b_servos_bottomleft.base.y2 = AUTO;
00209     b_servos_bottomleft.txtcolor = WHITE;
00210     b_servos_bottomleft.bgcolor = HEX(0xAE1010);
00211     b_servos_bottomleft.font = 0;
00212     b_servos_bottomleft.text = "BoLe";
00213     b_servos_bottomleft.callback = b_servos_bottomleft_cb;
00214     gui_button_add(&b_servos_bottomleft);
00215
00216     b_servos_bottomright.base.x1 = b_servos_bottomleft.base.x2 +
SERVO_BUTTON_SPACING;
00217     b_servos_bottomright.base.y1 = SERVO_BUTTON_Y - 3;
00218     b_servos_bottomright.base.x2 = AUTO;
00219     b_servos_bottomright.base.y2 = AUTO;
00220     b_servos_bottomright.txtcolor = WHITE;
00221     b_servos_bottomright.bgcolor = HEX(0xAE1010);
00222     b_servos_bottomright.font = 0;
00223     b_servos_bottomright.text = "BoRi";
00224     b_servos_bottomright.callback = b_servos_bottomright_cb;
00225     gui_button_add(&b_servos_bottomright);
00226
00227     //Led Color stuff
00228     #define LED_COLOR_BUTTON_Y 35
00229     #define LED_COLOR_BUTTON_SPACING 5
00230     tft_print_line(5, LED_COLOR_BUTTON_Y, BLACK,
TRANSPARENT, 0, "Led Color:");
00231
00232     b_led_off.base.x1 = 85;
00233     b_led_off.base.y1 = LED_COLOR_BUTTON_Y - 3;
00234     b_led_off.base.x2 = AUTO;
00235     b_led_off.base.y2 = AUTO;
00236     b_led_off.txtcolor = WHITE;
00237     b_led_off.bgcolor = BLACK;
00238

```

```

00239     b_led_off.font = 0;
00240     b_led_off.text = "Off";
00241     b_led_off.callback = b_led_off_cb;
00242     gui_button_add(&b_led_off);
00243
00244     b_led_white.base.x1 = b_led_off.base.x2 +
LED_COLOR_BUTTON_SPACING;
00245     b_led_white.base.y1 = LED_COLOR_BUTTON_Y - 3;
00246     b_led_white.base.x2 = AUTO;
00247     b_led_white.base.y2 = AUTO;
00248     b_led_white.txtcolor = BLACK;
00249     b_led_white.bgcolor = HEX(0xEEEEEE);
00250     b_led_white.font = 0;
00251     b_led_white.text = "White";
00252     b_led_white.callback = b_led_white_cb;
00253     gui_button_add(&b_led_white);
00254
00255     b_led_red.base.x1 = b_led_white.base.x2 +
LED_COLOR_BUTTON_SPACING;
00256     b_led_red.base.y1 = LED_COLOR_BUTTON_Y - 3;
00257     b_led_red.base.x2 = AUTO;
00258     b_led_red.base.y2 = AUTO;
00259     b_led_red.txtcolor = WHITE;
00260     b_led_red.bgcolor = HEX(0xEE0000);
00261     b_led_red.font = 0;
00262     b_led_red.text = "Red";
00263     b_led_red.callback = b_led_red_cb;
00264     gui_button_add(&b_led_red);
00265
00266     b_led_green.base.x1 = b_led_red.base.x2 +
LED_COLOR_BUTTON_SPACING;
00267     b_led_green.base.y1 = LED_COLOR_BUTTON_Y - 3;
00268     b_led_green.base.x2 = AUTO;
00269     b_led_green.base.y2 = AUTO;
00270     b_led_green.txtcolor = WHITE;
00271     b_led_green.bgcolor = HEX(0x00EE00);
00272     b_led_green.font = 0;
00273     b_led_green.text = "Green";
00274     b_led_green.callback = b_led_green_cb;
00275     gui_button_add(&b_led_green);
00276
00277     b_led_blue.base.x1 = b_led_green.base.x2 +
LED_COLOR_BUTTON_SPACING;
00278     b_led_blue.base.y1 = LED_COLOR_BUTTON_Y - 3;
00279     b_led_blue.base.x2 = AUTO;
00280     b_led_blue.base.y2 = AUTO;
00281     b_led_blue.txtcolor = WHITE;
00282     b_led_blue.bgcolor = HEX(0x0000EE);
00283     b_led_blue.font = 0;
00284     b_led_blue.text = "Blue";
00285     b_led_blue.callback = b_led_blue_cb;
00286     gui_button_add(&b_led_blue);
00287
00288     //Led MaxPower stuff
00289     #define LED_POWER_BUTTON_Y 70
00290     tft_print_line(5, LED_POWER_BUTTON_Y, BLACK,
TRANSPARENT, 0, "Led Maximum Current:");
00291
00292     //Num up down test
00293     n_led_powerlimit.x = 160;
00294     n_led_powerlimit.y = LED_POWER_BUTTON_Y - 7;
00295     n_led_powerlimit.fgcolor = WHITE;
00296     n_led_powerlimit.value = 10;
00297     n_led_powerlimit.max = 40;
00298     n_led_powerlimit.min = 0;
00299     n_led_powerlimit.callback = n_led_powerlimit_cb;
00300     gui_numupdown_add(&n_led_powerlimit);
00301
00302
00303
00304     state = detecting;
00305 }
00306
00307 static void leave(void* screen)
00308 {
00309     gui_button_remove(&b_back);
00310     gui_button_remove(&b_servos_center);
00311     gui_button_remove(&b_servos_topleft);
00312     gui_button_remove(&b_servos_topright);
00313     gui_button_remove(&b_servos_bottomleft);
00314     gui_button_remove(&b_servos_bottomright);
00315     gui_button_remove(&b_led_off);
00316     gui_button_remove(&b_led_white);
00317     gui_button_remove(&b_led_red);
00318     gui_button_remove(&b_led_green);
00319     gui_button_remove(&b_led_blue);
00320     gui_numupdown_remove(&n_led_powerlimit);

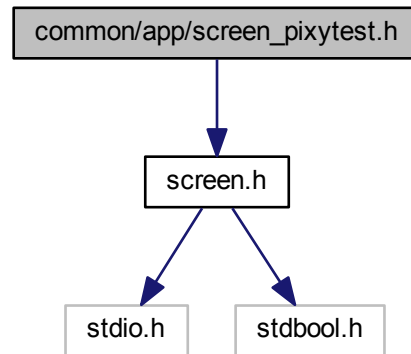
```

```
00321
00322 }
00323
00324
00325 static void update(void* screen)
00326 {
00327     switch (state) {
00328     case detecting: //Detecting State: Where we try to connect to the pixy
00329         if (pixy_init() == 0) { //Pixy connection ok
00330             int32_t response;
00331             int return_value;
00332             return_value = pixy_command("stop", END_OUT_ARGS, &response,
END_IN_ARGS);
00333             pixy_led_set_max_current(10);
00334
00335             state = idle; //Go to next state
00336         }
00337         break;
00338     case idle:
00339         pixy_service();
00340         break;
00341     case update_servos:
00342         pixy_rcs_set_position(0, servo_x);
00343         pixy_rcs_set_position(1, servo_y);
00344         state = idle;
00345         break;
00346     case update_ledcolor: {
00347         int32_t response;
00348         int return_value;
00349         return_value = pixy_command("led_set", INT32(led_color),
END_OUT_ARGS, &response, END_IN_ARGS);
00350         state = idle;
00351     }
00352     break;
00353     case update_ledcurrent:
00354         pixy_led_set_max_current(led_maxcurrent);
00355         state = idle;
00356         break;
00357     }
00358 }
00359
00360
00361 static SCREEN_STRUCT screen = {
00362     enter,
00363     leave,
00364     update
00365 };
00366
00367 SCREEN_STRUCT* get_screen_pixytest()
00368 {
00369     return &screen;
00370 }
```

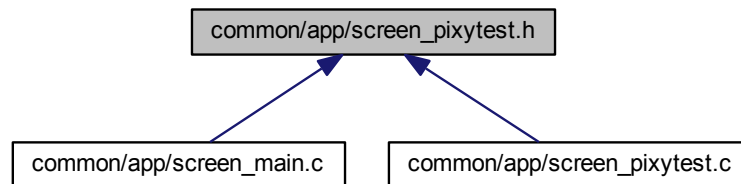
7.35 common/app/screen_pixytest.h File Reference

```
#include "screen.h"
```

Include dependency graph for screen_pixytest.h:



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



Functions

- `SCREEN_STRUCT * get_screen_pixytest ()`

7.36 screen_pixytest.h

```

00001 /*****
00002  * Project:      discoverpixy
00003  * Website:     https://github.com/t-moe/discoverpixy
00004  * Authors:     Aaron Schmocker, Timo Lang
00005  * Institution:  BFH Bern University of Applied Sciences
00006  * File:        common/app/screen_pixytest.h
00007  *
00008  * Version History:
00009  * Date          Autor Email          SHA      Changes
00010  * 2015-04-27    timolang@gmail.com    cf72baa  Introduced a Screen (sub) module and divided app into multiple
00011  * 2015-05-10    timolang@gmail.com    21edc56  Added doxyfile (doxygen) for the common folder. Started with
  
```



```

doxygen comments for app and tft module.
00012 * 2015-05-15 timolang@gmail.com 9a16865 Added doxygen comments to filesystem, checkbox, numupdown and
screen module. And some minor changes to the other modules.
00013 *
00014 *****/
00015
00016 #include "screen.h"
00017
00022
00028
00034 SCREEN_STRUCT* get_screen_pixytest();
00035

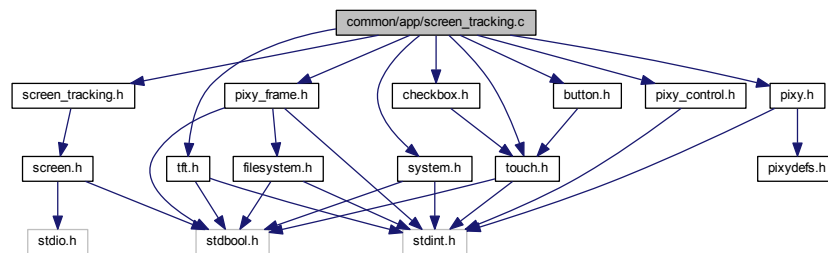
```

7.37 common/app/screen_tracking.c File Reference

```

#include "screen_tracking.h"
#include "pixy_control.h"
#include "button.h"
#include "checkbox.h"
#include "tft.h"
#include "touch.h"
#include "pixy.h"
#include "system.h"
#include "pixy_frame.h"
Include dependency graph for screen_tracking.c:

```



Data Structures

- struct [TRACKING_CONFIG_STRUCT](#)

Macros

- #define [FRAME_START_X](#) 1
- #define [FRAME_START_Y](#) 41
- #define [FRAME_WIDTH](#) 318
- #define [FRAME_HEIGHT](#) 198
- #define [FRAME_END_X](#) [FRAME_START_X](#) + [FRAME_WIDTH](#) - 1
- #define [FRAME_END_Y](#) [FRAME_START_Y](#) + [FRAME_HEIGHT](#) - 1
- #define [BLOCK_BUFFER_SIZE](#) 5

Typedefs

- typedef void(* [TRACKING_VOID_CALLBACK](#)) (void *tracking_config)
- typedef void(* [TRACKING_BLOCK_CALLBACK](#)) (void *tracking_config, struct [Block](#) *blocks, int num_↵ blocks)

Enumerations

- enum {
[detecting](#), [init](#), [tracking](#), [preselecting](#),
[abortselecting](#), [selecting](#), [selected](#), [error](#) }

Functions

- static void [b_back_cb](#) (void *button)
- static void [c_frame_toggle_cb](#) (void *checkbox, bool checked)
- static void [b_select_cb](#) (void *button)
- static void [touchCB](#) (void *touchArea, [TOUCH_ACTION](#) triggeredAction)
- void [tracking_our_start](#) (void *tracking_config)
- void [tracking_our_stop](#) (void *tracking_config)
- void [tracking_our_update](#) (void *tracking_config, struct [Block](#) *blocks, int num_blocks)
- void [tracking_reference_start](#) (void *tracking_config)
- void [tracking_reference_stop](#) (void *tracking_config)
- void [tracking_reference_update](#) (void *tracking_config, struct [Block](#) *blocks, int num_blocks)
- void [tracking_set_mode](#) (enum [Tracking_Implementation](#) impl)
- static void [enter](#) (void *screen)
- static void [leave](#) (void *screen)
- static void [update](#) (void *screen)
- [SCREEN_STRUCT](#) * [get_screen_tracking](#) ()

Variables

- static [BUTTON_STRUCT](#) [b_back](#)
- static [BUTTON_STRUCT](#) [b_select](#)
- static [CHECKBOX_STRUCT](#) [c_frame_toggle](#)
- static [TOUCH_AREA_STRUCT](#) [a_area](#)
- static volatile bool [frame_visible](#) = false
- static enum { ... } [state](#)
- static [POINT_STRUCT](#) [point1](#)
- static [POINT_STRUCT](#) [point2](#)
- static bool [point1_valid](#)
- static int16_t [servo_x](#) = 0
- static int16_t [servo_y](#) = 0
- static [TRACKING_CONFIG_STRUCT](#) [tracking_our](#)
- static [TRACKING_CONFIG_STRUCT](#) [tracking_reference](#)
- static [TRACKING_CONFIG_STRUCT](#) * [tracking_current](#)
- static [SCREEN_STRUCT](#) [screen](#)

7.37.1 Macro Definition Documentation

7.37.1.1 `#define BLOCK_BUFFER_SIZE 5`

7.37.1.2 `#define FRAME_END_X FRAME_START_X + FRAME_WIDTH - 1`

Definition at line 70 of file [screen_tracking.c](#).

7.37.1.3 `#define FRAME_END_Y FRAME_START_Y + FRAME_HEIGHT - 1`

Definition at line 71 of file [screen_tracking.c](#).

7.37.1.4 #define FRAME_HEIGHT 198

Definition at line 69 of file [screen_tracking.c](#).

7.37.1.5 #define FRAME_START_X 1

Definition at line 66 of file [screen_tracking.c](#).

7.37.1.6 #define FRAME_START_Y 41

Definition at line 67 of file [screen_tracking.c](#).

7.37.1.7 #define FRAME_WIDTH 318

Definition at line 68 of file [screen_tracking.c](#).

7.37.2 Typedef Documentation

7.37.2.1 typedef void(* TRACKING_BLOCK_CALLBACK) (void *tracking_config, struct Block *blocks, int num_blocks)

Definition at line 100 of file [screen_tracking.c](#).

7.37.2.2 typedef void(* TRACKING_VOID_CALLBACK) (void *tracking_config)

Definition at line 98 of file [screen_tracking.c](#).

7.37.3 Enumeration Type Documentation

7.37.3.1 anonymous enum

Enumerator

detecting

init

tracking

preselecting

abortselecting

selecting

selected

error

Definition at line 48 of file [screen_tracking.c](#).

```
00048 {detecting, init, tracking, preselecting,
      abortselecting, selecting, selected, error}
      state; //Current state of the screen state machine
```

7.37.4 Function Documentation

7.37.4.1 static void b_back_cb (void * *button*) [static]

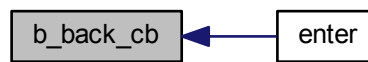
Definition at line 36 of file [screen_tracking.c](#).

```
00037 {  
00038     gui_screen_back(); //navigate back to the previous screen  
00039 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



7.37.4.2 static void b_select_cb (void * *button*) [static]

Definition at line 55 of file [screen_tracking.c](#).

```
00056 {  
00057     if (state == selecting) { //we're currently selecting a color region  
00058         state = abortselecting; //Abort selecting!!  
00059     } else if (state == tracking) { //we're currently watching the tracking  
00060         state = preselecting; //start selecting  
00061     }  
00062 }
```

Here is the caller graph for this function:

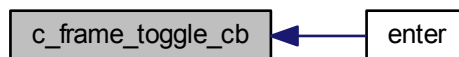


7.37.4.3 static void c_frame_toggle_cb (void * *checkbox*, bool *checked*) [static]

Definition at line 42 of file [screen_tracking.c](#).

```
00043 {
00044     frame_visible = checked; //Set the visibility of the frame to the checked state of the
checkbox
00045     //Frame will be drawn in the main loop below
00046 }
```

Here is the caller graph for this function:



7.37.4.4 static void enter (void * *screen*) [static]

Definition at line 229 of file [screen_tracking.c](#).

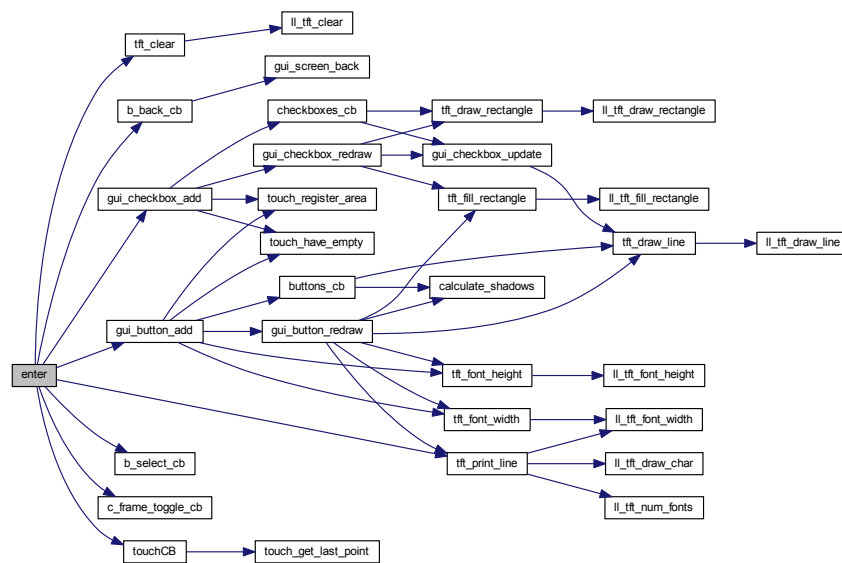
```
00230 {
00231     tft_clear(WHITE);
00232
00233     //"Back" button
00234     b_back.base.x1 = 5; //Start X of Button
00235     b_back.base.y1 = 5; //Start Y of Button
00236     b_back.base.x2 = AUTO; //Auto Calculate X2 with String Width
00237     b_back.base.y2 = AUTO; //Auto Calculate Y2 with String Height
00238     b_back.txtcolor = WHITE; //Set foreground color
00239     b_back.bgcolor = HEX(0xAE1010); //Set background color (Don't take 255 or 0 on at least
one channel, to make shadows possible)
00240     b_back.font = 0; //Select Font
00241     b_back.text = "Back"; //Set Text (For formatted strings take sprintf)
00242     b_back.callback = b_back_cb; //Call b_back_cb as Callback
00243     gui_button_add(&b_back); //Register Button (and run the callback from now on)
00244
00245
00246     //"Select color" button
00247     b_select.base.x1 = 150;
00248     b_select.base.y1 = 5;
00249     b_select.base.x2 = AUTO;
00250     b_select.base.y2 = AUTO;
00251     b_select.txtcolor = WHITE;
00252     b_select.bgcolor = HEX(0xAE1010);
00253     b_select.font = 0;
00254     b_select.text = "Select Color";
00255     b_select.callback = b_select_cb;
00256     gui_button_add(&b_select);
00257
00258     //"Frame visible" checkbox
00259     c_frame_toggle.base.x1 = 50;
00260     c_frame_toggle.base.x2 = 50 + 16;
00261     c_frame_toggle.base.y1 = 5;
00262     c_frame_toggle.base.y2 = 5 + 16;
00263     c_frame_toggle.checked = frame_visible;
00264     c_frame_toggle.fgcolor = CHECKBOX_WIN_FG_COLOR;
00265     c_frame_toggle.callback = c_frame_toggle_cb;
00266     gui_checkbox_add(&c_frame_toggle);
00267     tft_print_line(73, 8, BLACK, TRANSPARENT, 0, "Show Video");
00268
00269
00270     //Area to select a "color region"
00271     a_area.hookedActions = PEN_DOWN | PEN_UP;
00272     a_area.x1 = FRAME_START_X;
00273     a_area.y1 = FRAME_START_Y;
00274     a_area.x2 = FRAME_END_X;
00275     a_area.y2 = FRAME_END_Y;
}
```

```

00276     a_area.callback = touchCB;
00277     //Do not register it here, we do that later
00278
00279     if (tracking_current == NULL) {
00280         state = error;
00281     } else {
00282         state = detecting; //Start with the detecting state
00283     }
00284 }

```

Here is the call graph for this function:



7.37.4.5 static void leave (void * screen) [static]

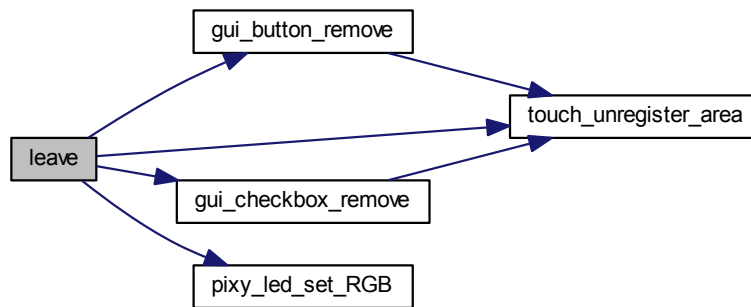
Definition at line 287 of file [screen_tracking.c](#).

```

00288 {
00289     //Remove buttons and checkbox
00290     gui_button_remove(&b_back);
00291     gui_button_remove(&b_select);
00292     gui_checkbox_remove(&c_frame_toggle);
00293
00294     if (state == selecting) { //the user left the screen in the "selecting" phase
00295         touch_unregister_area(&a_area); //remove the touch area
00296     }
00297
00298     if (state == tracking) { //the user left the screen in the "tracking" phase
00299         tracking_current->stop(tracking_current); //stop tracking
00300         pixy_led_set_RGB(0, 0, 0);
00301     }
00302 }

```

Here is the call graph for this function:



7.37.4.6 static void touchCB (void * touchArea, TOUCH_ACTION triggeredAction) [static]

Definition at line 75 of file [screen_tracking.c](#).

```

00076 {
00077     POINT_STRUCT p = touch_get_last_point();
00078
00079     switch (triggeredAction) {
00080     case PEN_DOWN: //The user just put down the pen
00081         point1.x = p.x - FRAME_START_X; //Calculate x-Coordinate relative to frame
00082         point1.y = p.y - FRAME_START_Y; //Calculate y-Coordinate relative to frame
00083         point1_valid = true; //The point1 is now valid
00084         break;
00085
00086     case PEN_UP: //The user took the pen away
00087         if (point1_valid) { //only execute if point1 is valid
00088             point2.x = p.x - FRAME_START_X; //Calculate x-Coordinate relative to frame
00089             point2.y = p.y - FRAME_START_Y; //Calculate y-Coordinate relative to frame
00090             state = selected;
00091         }
00092         break;
00093     }
00094 }
00095 }
  
```

Here is the call graph for this function:



Here is the caller graph for this function:



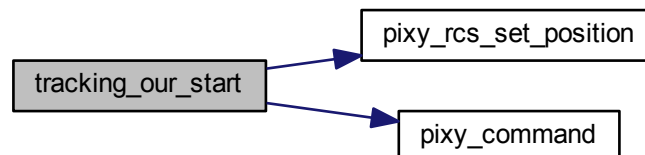
7.37.4.7 void tracking_our_start (void * tracking_config)

Definition at line 114 of file [screen_tracking.c](#).

```

00115 {
00116     //Activate pixy's data send program
00117     int32_t response;
00118     int return_value;
00119
00120     servo_x = servo_y = 500;           // set a default value of 500
00121     pixy_rcs_set_position(0, servo_x); // set default
00122     pixy_rcs_set_position(1, servo_y); // set default
00123
00124     return_value = pixy_command("runprog", INT8(0), END_OUT_ARGS, &response,
00125                                END_IN_ARGS);
00126 }
  
```

Here is the call graph for this function:



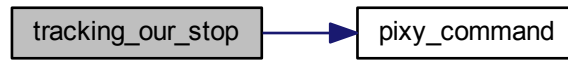
7.37.4.8 void tracking_our_stop (void * tracking_config)

Definition at line 128 of file [screen_tracking.c](#).

```

00129 {
00130     //Stop pixy's data send programm
00131     int32_t response;
00132     int return_value;
00133     return_value = pixy_command("stop", END_OUT_ARGS, &response,
00134                                END_IN_ARGS);
00135 }
  
```


Here is the call graph for this function:



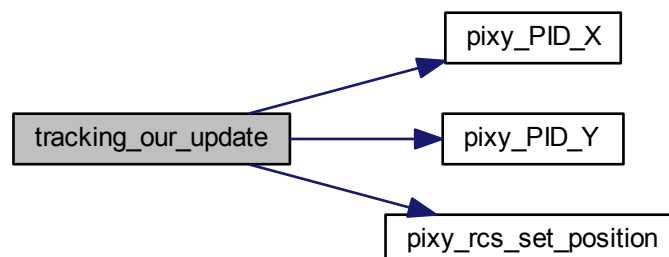
7.37.4.9 void tracking_our_update (void * tracking_config, struct Block * blocks, int num_blocks)

Definition at line 137 of file [screen_tracking.c](#).

```

00138 {
00139
00140     if (num_blocks <= 0) {                // Check if there are blocks available
00141         return;                          // When there are none, do nothing
00142     }
00143
00144     uint16_t x = blocks[0].x;             // Get x coordinate of the biggest object
00145     uint16_t y = blocks[0].y;             // Get y coordinate of the biggest object
00146
00147     int16_t xset = 0;
00148     int16_t yset = 0;
00149
00150     xset = (servo_x + pixy_PID_X((FRAME_WIDTH / 2), x)); // calculate the
PID output for x
00151     yset = (servo_y - pixy_PID_Y((FRAME_HEIGHT / 2), y)); // calculate the
PID output for y
00152
00153     xset = (xset < 0) ? 0 : xset;          // x lower boundary check
00154     xset = (xset > 1000) ? 1000 : xset;    // x upper boundary check
00155
00156     yset = (yset < 0) ? 0 : yset;          // y lower boundary check
00157     yset = (yset > 1000) ? 1000 : yset;    // y upper boundary check
00158
00159     servo_x = xset;                       // update the global, static variable for x
00160     servo_y = yset;                       // update the global, static variable for y
00161
00162     pixy_rcs_set_position(0, servo_x);    // set the new x position
00163     pixy_rcs_set_position(1, servo_y);    // set the new y position
00164 }
  
```

Here is the call graph for this function:

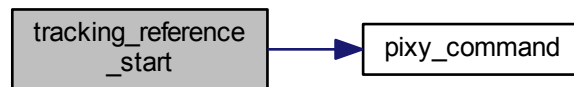


7.37.4.10 void tracking_reference_start (void * *tracking_config*)

Definition at line 176 of file [screen_tracking.c](#).

```
00177 {  
00178     //Run reference tracking  
00179     int32_t response;  
00180     int return_value;  
00181     return_value = pixy_command("runprog", INT8(2), END_OUT_ARGS, &response,  
                                END_IN_ARGS);  
00182 }
```

Here is the call graph for this function:



7.37.4.11 void tracking_reference_stop (void * *tracking_config*)

Definition at line 185 of file [screen_tracking.c](#).

```
00186 {  
00187     //Stop reference tracking  
00188     int32_t response;  
00189     int return_value;  
00190     return_value = pixy_command("stop", END_OUT_ARGS, &response,  
                                END_IN_ARGS);  
00191 }
```

Here is the call graph for this function:



7.37.4.12 void tracking_reference_update (void * *tracking_config*, struct Block * *blocks*, int *num_blocks*)

Definition at line 194 of file [screen_tracking.c](#).

```
00195 {  
00196     //Nothing to do here. Pixy does it all.  
00197 }
```

7.37.4.13 static void update (void * screen) [static]

Definition at line 306 of file screen_tracking.c.

```

00307 {
00308     switch (state) {
00309     case detecting: //Detecting State: Where we try to connect to the pixy
00310         if (pixy_init() == 0) { //Pixy connection ok
00311             state = init; //Go to next state
00312         }
00313     }
00314     break;
00315
00316     case init: //Init State: Where we start the tracking
00317         tracking_current->start(tracking_current);
00318         state = tracking;
00319         break;
00320
00321     case tracking: //Tracking state: Where we render the frame and the tracked objects
00322         pixy_service(); //Receive events (e.g. block-data) from pixy
00323
00324         if (pixy_blocks_are_new()) { //There are new blocks available
00325             if (frame_visible) { //If the user want's us to draw the video data
00326                 pixy_render_full_frame(FRAME_START_X,
FRAME_START_Y);
00327             } else { //the user want's a colored background
00328                 tft_fill_rectangle(FRAME_START_X,
FRAME_START_Y, FRAME_END_X, FRAME_END_Y, RGB(200, 200, 200));
00329             }
00330
00331             #define BLOCK_BUFFER_SIZE 5 //The maximum amount of blocks that we want to receive
00332             struct Block blocks[BLOCK_BUFFER_SIZE]; //Storage to receive blocks from
pixy
00333             int blocks_received = pixy_get_blocks(
BLOCK_BUFFER_SIZE, blocks); //Try to receive up to BLOCK_BUFFER_SIZE Blocks from pixy
00334
00335             if (blocks_received >= 0) { //block receiving ok
00336                 tracking_current->update(tracking_current, blocks,
blocks_received); //apply tracking
00337
00338                 //Draw blocks
00339                 for (int i = 0; i < blocks_received; i++) { //for each received block
00340                     struct Block* block = &(blocks[i]);
00341                     //block.x and block.y are the center coordinates of the object relative to the camera
origin.
00342                     uint16_t x = block->x - 1 + FRAME_START_X - block->
width / 2; //Calculate x-Coordinate on the display
00343                     uint16_t y = block->y - 1 + FRAME_START_Y - block->
height / 2; //Calculate y-Coordinate on the display
00344                     tft_draw_rectangle(x, y, x + block->width - 1, y + block->
height - 1, WHITE); //Draw a white rectangle
00345                 }
00346             }
00347         }
00348         break;
00349
00350     case preselecting: { //Pre-Selecting State: Where we set up the color region selection
00351         tracking_current->stop(tracking_current); //Stop tracking
00352
00353         pixy_render_full_frame(FRAME_START_X,
FRAME_START_Y); //Render one frame
00354
00355         touch_register_area(&a_area); //Register touch area and receive events
from now on
00356         point1_valid = false; //we start with an invalid point1
00357
00358         b_select.text = "Abort"; //Change the button text to "Abort"
00359         gui_button_redraw(&b_select); //redraw button
00360
00361         state = selecting; //The user can now select a region
00362     }
00363     break;
00364
00365     case selected: { //Selected State: Where we send the users selection to pixy
00366         //Ensure that (x1,y1) represent the top-left point and (x2,y2) the bottom-right.
00367         unsigned int tmp;
00368
00369         if (point1.x > point2.x) {
00370             tmp = point1.x;
00371             point1.x = point2.x;
00372             point2.x = tmp;
00373         }
00374
00375         if (point1.y > point2.y) {
00376             tmp = point1.y;
00377             point1.y = point2.y;
00378             point2.y = tmp;
00379         }
00380     }
00381     break;
00382 }

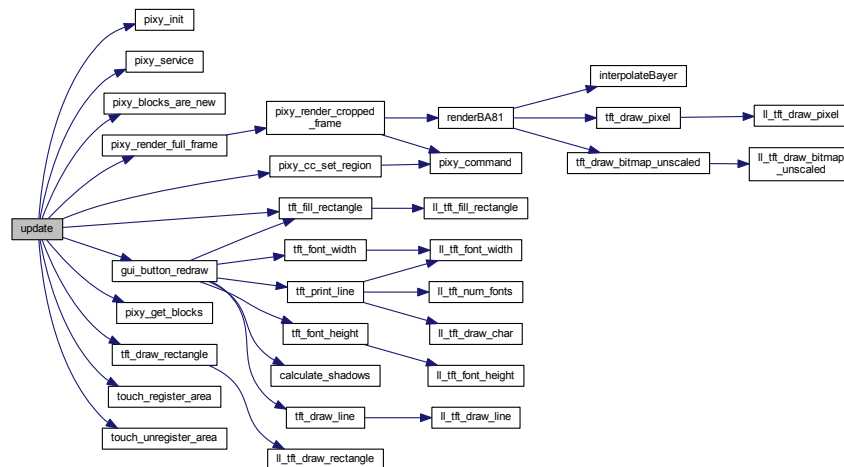
```

```

00378         point1.y = point2.y;
00379         point2.y = tmp;
00380     }
00381
00382     //Send pixy the selected region
00383     pixy_cc_set_region(1, point1.x, point1.y,
00384         point2.x - point1.x, point2.y - point1.y);
00385 }
00386
00387 //no break here: We want the following code to be executed as well
00388 case abortselecting: { //Abort-Selecting State: Where we deinitialize the stuff we used
00389     for region selection
00390     touch_unregister_area(&a_area); //Remove the touch area. We'll no longer
00391     receive touch events
00392
00393     b_select.text = "Select Color"; //Change the button text back to "Select Color"
00394     gui_button_redraw(&b_select); //redraw button
00395
00396     tracking_current->start(tracking_current); //Start tracking
00397
00398     again
00399     state = tracking;
00400 }
00401 break;
00402
00403 case selecting: //Selecting State: Where we wait on the user to select a color region
00404     pixy_service(); //receive pixy events
00405     //wait on user to select the image area
00406     break;
00407
00408 case error: //Error State: Where we show an error message and leave the user no other choice than
00409     to click the backbutton
00410     //wait on user to click the back button
00411     break;
00412 }
00413 }
00414 }

```

Here is the call graph for this function:



7.37.5 Variable Documentation

7.37.5.1 TOUCH_AREA_STRUCT a_area [static]

Definition at line 33 of file [screen_tracking.c](#).

7.37.5.2 BUTTON_STRUCT b_back [static]

Definition at line 30 of file [screen_tracking.c](#).

7.37.5.3 BUTTON_STRUCT b_select [static]

Definition at line 31 of file [screen_tracking.c](#).

7.37.5.4 CHECKBOX_STRUCT c_frame_toggle [static]

Definition at line 32 of file [screen_tracking.c](#).

7.37.5.5 volatile bool frame_visible = false [static]

Definition at line 41 of file [screen_tracking.c](#).

7.37.5.6 POINT_STRUCT point1 [static]

Definition at line 50 of file [screen_tracking.c](#).

7.37.5.7 bool point1_valid [static]

Definition at line 52 of file [screen_tracking.c](#).

7.37.5.8 POINT_STRUCT point2 [static]

Definition at line 51 of file [screen_tracking.c](#).

7.37.5.9 SCREEN_STRUCT screen [static]

Initial value:

```
= {  
    enter,  
    leave,  
    update  
}
```

Definition at line 411 of file [screen_tracking.c](#).

7.37.5.10 int16_t servo_x = 0 [static]

Definition at line 110 of file [screen_tracking.c](#).

7.37.5.11 int16_t servo_y = 0 [static]

Definition at line 111 of file [screen_tracking.c](#).

7.37.5.12 enum { ... } state [static]**7.37.5.13 TRACKING_CONFIG_STRUCT* tracking_current** [static]

Definition at line 207 of file [screen_tracking.c](#).

7.37.5.14 TRACKING_CONFIG_STRUCT tracking_our [static]

Initial value:

```
= {
    tracking_our_start,
    tracking_our_stop,
    tracking_our_update
}
```

Definition at line 167 of file [screen_tracking.c](#).

7.37.5.15 TRACKING_CONFIG_STRUCT tracking_reference [static]

Initial value:

```
= {
    tracking_reference_start,
    tracking_reference_stop,
    tracking_reference_update
}
```

Definition at line 200 of file [screen_tracking.c](#).

7.38 screen_tracking.c

```
00001 /*****
00002  * Project:      discoverpixy
00003  * Website:     https://github.com/t-moe/discoverpixy
00004  * Authors:     Aaron Schmocker, Timo Lang
00005  * Institution: BFH Bern University of Applied Sciences
00006  * File:        common/app/screen_tracking.c
00007  *
00008  * Version History:
00009  * Date          Autor Email          SHA          Changes
00010  * 2015-05-16    timolang@gmail.com    e46314b Added Tracking Screen and implemented "Reference Tracking" and
00011  *               timolang@gmail.com    8088014 Updated Tracking Screen so that the implementations are
00012  *               aaron@duckpond.ch      8c264c2 Comment refactoring, updated PID values
00013  *               aaron@duckpond.ch      a04cda9 Refactured comments and implemented a bugfix for the PID
00014  *               aaron@duckpond.ch      802d3df Fixed pid controller and refactored code
00015  *               aaron@duckpond.ch      3d98ca9 Minor changes
00016  *               timolang@gmail.com     c87220d Renamed pixy_helper to pixy_frame. Updated docu of appliaction.
00017  *               added doxygen comments to pixy_{frame,control}.h
00018  *****/
00019
00020 #include "screen_tracking.h"
00021 #include "pixy_control.h"
00022 #include "button.h"
00023 #include "checkbox.h"
00024 #include "tft.h"
00025 #include "touch.h"
00026 #include "pixy.h"
00027 #include "system.h"
00028 #include "pixy_frame.h"
00029
00030 static BUTTON_STRUCT b_back; //Button to navigate back
00031 static BUTTON_STRUCT b_select; //Button to start the color region selection
00032 static CHECKBOX_STRUCT c_frame_toggle; //Checkbox to toggle video data on/off
00033 static TOUCH_AREA_STRUCT a_area; //Touch area for the color region selection
00034
00035 //Callback for when the user presses the "back" button
00036 static void b_back_cb(void* button)
00037 {
00038     gui_screen_back(); //navigate back to the previous screen
00039 }
00040
00041 static volatile bool frame_visible = false; //Whether or not the video data should be
00042         displayed
```

```

00042 static void c_frame_toggle_cb(void* checkbox, bool checked)
00043 {
00044     frame_visible = checked; //Set the visibility of the frame to the checked state of the
        checkbox
00045     //Frame will be drawn in the main loop below
00046 }
00047
00048 static enum {detecting, init, tracking, preselecting,
        abortselecting, selecting, selected, error}
        state; //Current state of the screen state machine
00049
00050 static POINT_STRUCT point1; //First point of the rectangle selected by the user (color
        region selection)
00051 static POINT_STRUCT point2; //End point of the rectangle selected by the user (color
        region selection)
00052 static bool point1_valid; //Whether or not we have a valid first point
00053
00054 //Callback for when the user presses the "select color" button
00055 static void b_select_cb(void* button)
00056 {
00057     if (state == selecting) { //we're currently selecting a color region
00058         state = abortselecting; //Abort selecting!!
00059     } else if (state == tracking) { //we're currently watching the tracking
00060         state = preselecting; //start selecting
00061     }
00062 }
00063
00064 //Video Region properties
00065 //The camera records with 320*200px, but we need to keep a 1px border because of color interpolation (bayer
        format)
00066 #define FRAME_START_X 1 //x-Coordinate of the top-left point of the frame rectangle on display
00067 #define FRAME_START_Y 41 //y-Coordinate of the top-left point of the frame rectangle on display
00068 #define FRAME_WIDTH 318 //Width of the video frame
00069 #define FRAME_HEIGHT 198 //Height of the video frame
00070 #define FRAME_END_X FRAME_START_X +FRAME_WIDTH-1 //x-Coordinate of the bottom-right point of the frame
        rectangle
00071 #define FRAME_END_Y FRAME_START_Y +FRAME_HEIGHT-1 //y-Coordinate of the bottom-right point of the frame
        rectangle
00072
00073 //Callback for when the user touches the frame area to select a color region.
00074 //Note: It doesn't matter in which direction the user draws the rectangle, we'll normalize the coordinates
        later
00075 static void touchCB(void* touchArea, TOUCH_ACTION triggeredAction)
00076 {
00077     POINT_STRUCT p = touch_get_last_point();
00078
00079     switch (triggeredAction) {
00080     case PEN_DOWN: //The user just put down the pen
00081         point1.x = p.x - FRAME_START_X; //Calculate x-Coordinate relative to frame start
00082         point1.y = p.y - FRAME_START_Y; //Calculate y-Coordinate relative to frame start
00083         point1_valid = true; //The point1 is now valid
00084         break;
00085
00086     case PEN_UP: //The user took the pen away
00087         if (point1_valid) { //only execute if point1 is valid
00088             point2.x = p.x - FRAME_START_X; //Calculate x-Coordinate relative to frame start
00089             point2.y = p.y - FRAME_START_Y; //Calculate y-Coordinate relative to frame start
00090             state = selected;
00091         }
00092
00093         break;
00094     }
00095 }
00096
00097 //Prototype for tracking start/stop methods
00098 typedef void (*TRACKING_VOID_CALLBACK)(void* tracking_config);
00099 //Prototype for tracking update method
00100 typedef void (*TRACKING_BLOCK_CALLBACK)(void* tracking_config, struct
        Block* blocks, int num_blocks);
00101
00102 //Structure to save callbacks and settings of a tracking implementation
00103 typedef struct {
00104     TRACKING_VOID_CALLBACK start;
00105     TRACKING_VOID_CALLBACK stop;
00106     TRACKING_BLOCK_CALLBACK update;
00107 } TRACKING_CONFIG_STRUCT;
00108
00109 //Methods for our tracking implementation ahead
00110 static int16_t servo_x = 0;
00111 static int16_t servo_y = 0;
00112
00113 //Method/Callback to start our tracking
00114 void tracking_our_start(void* tracking_config)
00115 {
00116     //Activate pixy's data send program
00117     int32_t response;
00118     int return_value;

```

```

00119
00120     servo_x = servo_y = 500;           // set a default value of 500
00121     pixy_rcs_set_position(0, servo_x); // set default
00122     pixy_rcs_set_position(1, servo_y); // set default
00123
00124     return_value = pixy_command("runprog", INT8(0), END_OUT_ARGS, &response,
END_IN_ARGS);
00125 }
00126
00127 //Method/Callback to stop our tracking
00128 void tracking_our_stop(void* tracking_config)
00129 {
00130     //Stop pixy's data send programm
00131     int32_t response;
00132     int return_value;
00133     return_value = pixy_command("stop", END_OUT_ARGS, &response,
END_IN_ARGS);
00134 }
00135
00136 //Method/Callback to calculate one step of our tracking
00137 void tracking_our_update(void* tracking_config, struct Block* blocks, int
num_blocks)
00138 {
00139
00140     if (num_blocks <= 0) {             // Check if there are blocks available
00141         return;                       // When there are none, do nothing
00142     }
00143
00144     uint16_t x = blocks[0].x;          // Get x coordinate of the biggest object
00145     uint16_t y = blocks[0].y;          // Get y coordinate of the biggest object
00146
00147     int16_t xset = 0;
00148     int16_t yset = 0;
00149
00150     xset = (servo_x + pixy_PID_X((FRAME_WIDTH / 2), x)); // calculate the
PID output for x
00151     yset = (servo_y - pixy_PID_Y((FRAME_HEIGHT / 2), y)); // calculate the
PID output for y
00152
00153     xset = (xset < 0) ? 0 : xset;      // x lower boundary check
00154     xset = (xset > 1000) ? 1000 : xset; // x upper boundary check
00155
00156     yset = (yset < 0) ? 0 : yset;      // y lower boundary check
00157     yset = (yset > 1000) ? 1000 : yset; // y upper boundary check
00158
00159     servo_x = xset;                   // update the global, static variable for x
00160     servo_y = yset;                   // update the global, static variable for y
00161
00162     pixy_rcs_set_position(0, servo_x); // set the new x position
00163     pixy_rcs_set_position(1, servo_y); // set the new y position
00164 }
00165
00166 //Variable which stores all the callbacks and settings for our tracking implementation
00167 static TRACKING_CONFIG_STRUCT tracking_our = {
00168     tracking_our_start,
00169     tracking_our_stop,
00170     tracking_our_update
00171 };
00172
00173 //Methods for reference tracking implementation ahead
00174
00175 //Method/Callback to start reference tracking
00176 void tracking_reference_start(void* tracking_config)
00177 {
00178     //Run reference tracking
00179     int32_t response;
00180     int return_value;
00181     return_value = pixy_command("runprog", INT8(2), END_OUT_ARGS, &response,
END_IN_ARGS);
00182 }
00183
00184 //Method/Callback to stop reference tracking
00185 void tracking_reference_stop(void* tracking_config)
00186 {
00187     //Stop reference tracking
00188     int32_t response;
00189     int return_value;
00190     return_value = pixy_command("stop", END_OUT_ARGS, &response,
END_IN_ARGS);
00191 }
00192
00193 //Method/Callback to calculate one step of the reference tracking
00194 void tracking_reference_update(void* tracking_config, struct
Block* blocks, int num_blocks)
00195 {
00196     //Nothing to do here. Pixy does it all.
00197 }

```



```

00198
00199 //Variable which stores all the callbacks and settings for the reference tracking implementation
00200 static TRACKING_CONFIG_STRUCT tracking_reference = {
00201     tracking_reference_start,
00202     tracking_reference_stop,
00203     tracking_reference_update
00204 };
00205
00206 //Pointer to the currently active tracking implementation. See also tracking_set_mode
00207 static TRACKING_CONFIG_STRUCT* tracking_current;
00208
00209 //Method to set the current tracking implementation. This function is exported and should be called before
    getting the screen
00210 void tracking_set_mode(enum Tracking_Implementation impl)
00211 {
00212     //Depending on the enum value let tracking_current point to a different setting/callback structure
00213     switch (impl) {
00214         case OUR_TRACKING:
00215             tracking_current = &tracking_our;
00216             break;
00217
00218         case REFERENCE_TRACKING:
00219             tracking_current = &tracking_reference;
00220             break;
00221
00222         default:
00223             tracking_current = NULL;
00224             break;
00225     }
00226 }
00227
00228 //Callback for when the screen is entered/loaded
00229 static void enter(void* screen)
00230 {
00231     tft_clear(WHITE);
00232
00233     //"Back" button
00234     b_back.base.x1 = 5; //Start X of Button
00235     b_back.base.y1 = 5; //Start Y of Button
00236     b_back.base.x2 = AUTO; //Auto Calculate X2 with String Width
00237     b_back.base.y2 = AUTO; //Auto Calculate Y2 with String Height
00238     b_back.txtcolor = WHITE; //Set foreground color
00239     b_back.bgcolor = HEX(0xAE1010); //Set background color (Don't take 255 or 0 on at least one
channel, to make shadows possible)
00240     b_back.font = 0; //Select Font
00241     b_back.text = "Back"; //Set Text (For formatted strings take sprintf)
00242     b_back.callback = b_back_cb; //Call b_back_cb as Callback
00243     gui_button_add(&b_back); //Register Button (and run the callback from now on)
00244
00245     //"Select color" button
00246     b_select.base.x1 = 150;
00247     b_select.base.y1 = 5;
00248     b_select.base.x2 = AUTO;
00249     b_select.base.y2 = AUTO;
00250     b_select.txtcolor = WHITE;
00251     b_select.bgcolor = HEX(0xAE1010);
00252     b_select.font = 0;
00253     b_select.text = "Select Color";
00254     b_select.callback = b_select_cb;
00255     gui_button_add(&b_select);
00256
00257     //"Frame visible" checkbox
00258     c_frame_toggle.base.x1 = 50;
00259     c_frame_toggle.base.x2 = 50 + 16;
00260     c_frame_toggle.base.y1 = 5;
00261     c_frame_toggle.base.y2 = 5 + 16;
00262     c_frame_toggle.checked = frame_visible;
00263     c_frame_toggle.fgcolor = CHECKBOX_WIN_FG_COLOR;
00264     c_frame_toggle.callback = c_frame_toggle_cb;
00265     gui_checkbox_add(&c_frame_toggle);
00266     tft_print_line(73, 8, BLACK, TRANSPARENT, 0, "Show Video");
00267
00268
00269
00270     //Area to select a "color region"
00271     a_area.hookedActions = PEN_DOWN | PEN_UP;
00272     a_area.x1 = FRAME_START_X;
00273     a_area.y1 = FRAME_START_Y;
00274     a_area.x2 = FRAME_END_X;
00275     a_area.y2 = FRAME_END_Y;
00276     a_area.callback = touchCB;
00277     //Do not register it here, we do that later
00278
00279     if (tracking_current == NULL) {
00280         state = error;
00281     } else {
00282         state = detecting; //Start with the detecting state

```

```

00283     }
00284 }
00285
00286 //Callback for when the screen is left/unloaded
00287 static void leave(void* screen)
00288 {
00289     //Remove buttons and checkbox
00290     gui_button_remove(&b_back);
00291     gui_button_remove(&b_select);
00292     gui_checkbox_remove(&c_frame_toggle);
00293
00294     if (state == selecting) { //the user left the screen in the "selecting" phase
00295         touch_unregister_area(&a_area); //remove the touch area
00296     }
00297
00298     if (state == tracking) { //the user left the screen in the "tracking" phase
00299         tracking_current->stop(tracking_current); //stop tracking
00300         pixy_led_set_RGB(0, 0, 0);
00301     }
00302 }
00303
00304 //Callback for when the screen should be updated
00305 //This is the main loop of the screen. This method will be called repeatedly
00306 static void update(void* screen)
00307 {
00308     switch (state) {
00309     case detecting: //Detecting State: Where we try to connect to the pixy
00310         if (pixy_init() == 0) { //Pixy connection ok
00311             state = init; //Go to next state
00312         }
00313         break;
00314
00315     case init: //Init State: Where we start the tracking
00316         tracking_current->start(tracking_current);
00317         state = tracking;
00318         break;
00319
00320     case tracking: //Tracking state: Where we render the frame and the tracked objects
00321         pixy_service(); //Receive events (e.g. block-data) from pixy
00322
00323         if (pixy_blocks_are_new()) { //There are new blocks available
00324             if (frame_visible) { //If the user want's us to draw the video data
00325                 pixy_render_full_frame(FRAME_START_X,
00326                     FRAME_START_Y);
00327             } else { //the user want's a colored background
00328                 tft_fill_rectangle(FRAME_START_X,
00329                     FRAME_START_Y, FRAME_END_X, FRAME_END_Y, RGB(200, 200, 200));
00330             }
00331
00332             #define BLOCK_BUFFER_SIZE 5 //The maximum amount of blocks that we want to receive
00333             struct Block blocks[BLOCK_BUFFER_SIZE]; //Storage to receive blocks from pixy
00334             int blocks_received = pixy_get_blocks(BLOCK_BUFFER_SIZE, blocks); //Try to
00335             //receive up to BLOCK_BUFFER_SIZE Blocks from pixy
00336
00337             if (blocks_received >= 0) { //block receiving ok
00338                 tracking_current->update(tracking_current, blocks, blocks_received); //apply tracking
00339
00340                 //Draw blocks
00341                 for (int i = 0; i < blocks_received; i++) { //for each received block
00342                     struct Block* block = &(blocks[i]);
00343                     //block.x and block.y are the center coordinates of the object relative to the camera
00344                     origin.
00345                     uint16_t x = block->x - 1 + FRAME_START_X - block->
00346                     width / 2; //Calculate x-Coordinate on the display
00347                     uint16_t y = block->y - 1 + FRAME_START_Y - block->
00348                     height / 2; //Calculate y-Coordinate on the display
00349                     tft_draw_rectangle(x, y, x + block->width - 1, y + block->
00350                     height - 1, WHITE); //Draw a white rectangle
00351                 }
00352             }
00353             break;
00354
00355     case preselecting: { //Pre-Selecting State: Where we set up the color region selection
00356         tracking_current->stop(tracking_current); //Stop tracking
00357         pixy_render_full_frame(FRAME_START_X,
00358             FRAME_START_Y); //Render one frame
00359         touch_register_area(&a_area); //Register touch area and receive events from now
00360         on
00361         point1_valid = false; //we start with an invalid point1
00362         b_select.text = "Abort"; //Change the button text to "Abort"
00363         gui_button_redraw(&b_select); //redraw button

```

```

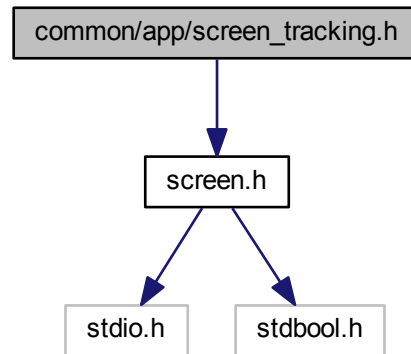
00361
00362     state = selecting; //The user can now select a region
00363 }
00364 break;
00365
00366 case selected: { //Selected State: Where we send the users selection to pixy
00367     //Ensure that (x1,y1) represent the top-left point and (x2,y2) the bottom-right.
00368     unsigned int tmp;
00369
00370     if (point1.x > point2.x) {
00371         tmp = point1.x;
00372         point1.x = point2.x;
00373         point2.x = tmp;
00374     }
00375
00376     if (point1.y > point2.y) {
00377         tmp = point1.y;
00378         point1.y = point2.y;
00379         point2.y = tmp;
00380     }
00381
00382     //Send pixy the selected region
00383     pixy_cc_set_region(1, point1.x, point1.y, point2.x - point1.
00384 x, point2.y - point1.y);
00385 }
00386
00387 //no break here: We want the following code to be executed as well
00388
00389 case abortselecting: { //Abort-Selecting State: Where we deinitialize the stuff we used
00390     for region selection
00391     touch_unregister_area(&a_area); //Remove the touch area. We'll no longer
00392     receive touch events
00393
00394     b_select.text = "Select Color"; //Change the button text back to "Select Color"
00395     gui_button_redraw(&b_select); //redraw button
00396
00397     tracking_current->start(tracking_current); //Start tracking again
00398     state = tracking;
00399 }
00400 break;
00401
00402 case selecting: //Selecting State: Where we wait on the user to select a color region
00403     pixy_service(); //receive pixy events
00404     //wait on user to select the image area
00405     break;
00406
00407 case error: //Error State: Where we show an error message and leave the user no other choice than
00408     to click the backbutton
00409     //wait on user to click the back button
00410     break;
00411 }
00412
00413 //Declare screen callbacks
00414 static SCREEN_STRUCT screen = {
00415     enter,
00416     leave,
00417     update
00418 };
00419
00420 SCREEN_STRUCT* get_screen_tracking()
00421 {
00422     return &screen;
00423 }

```

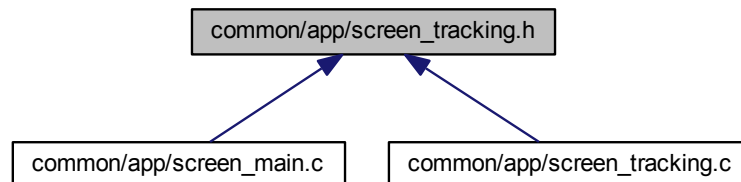
7.39 common/app/screen_tracking.h File Reference

```
#include "screen.h"
```

Include dependency graph for screen_tracking.h:



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



Enumerations

- enum `Tracking_Implementation` { `OUR_TRACKING`, `REFERENCE_TRACKING` }

Functions

- void `tracking_set_mode` (enum `Tracking_Implementation` impl)
- `SCREEN_STRUCT` * `get_screen_tracking` ()

7.40 screen_tracking.h

```

00001 /*****
00002  * Project:      discoverpixy
00003  * Website:     https://github.com/t-moe/discoverpixy

```

```

00004 * Authors:      Aaron Schmocker, Timo Lang
00005 * Institution:   BFH Bern University of Applied Sciences
00006 * File:         common/app/screen_tracking.h
00007 *
00008 * Version History:
00009 * Date          Autor Email          SHA          Changes
00010 * 2015-05-16    timolang@gmail.com    e46314b      Added Tracking Screen and implemented "Reference Tracking" and
               "Color Region Selection"
00011 *
00012 *****
00013 *****/
00013
00014 #include "screen.h"
00015
00020
00026
00027
00031 enum Tracking_Implementation {
00032     OUR_TRACKING,
00033     REFERENCE_TRACKING
00034 };
00035
00040 void tracking_set_mode(enum Tracking_Implementation impl);
00041
00047 SCREEN_STRUCT* get_screen_tracking();
00048

```

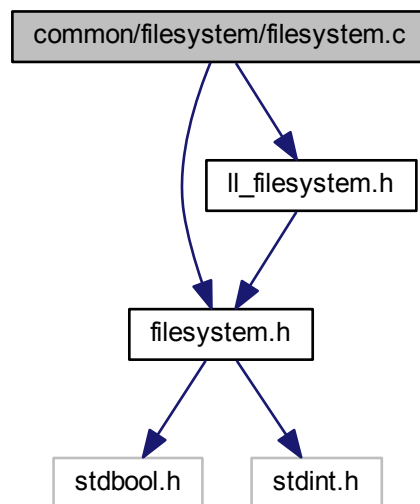
7.41 common/filesystem/filesystem.c File Reference

```

#include "filesystem.h"
#include "ll_filesystem.h"

```

Include dependency graph for filesystem.c:



Functions

- `bool filesystem_init ()`
- `DIRECTORY_STRUCT * filesystem_dir_open (const char *path)`
- `void filesystem_dir_close (DIRECTORY_STRUCT *dir)`
- `FILE_HANDLE * filesystem_file_open (const char *filename)`
- `void filesystem_file_close (FILE_HANDLE *handle)`

- `FILE_STATUS filesystem_file_seek (FILE_HANDLE *handle, uint32_t offset)`
- `FILE_STATUS filesystem_file_read (FILE_HANDLE *handle, uint8_t *buf, uint32_t size)`
- `FILE_STATUS filesystem_file_write (FILE_HANDLE *handle, uint8_t *buf, uint32_t size)`

7.42 filesystem.c

```

00001 /*****
00002  * Project:      discoverpixy
00003  * Website:     https://github.com/t-moe/discoverpixy
00004  * Authors:     Aaron Schmocker, Timo Lang
00005  * Institution: BFH Bern University of Applied Sciences
00006  * File:       common/filesystem/filesystem.c
00007  *
00008  * Version History:
00009  * Date      Autor Email      SHA      Changes
00010  * 2015-05-10 timolang@gmail.com e2bce8f Added filesystem module, tests and implementation for it in
00011  * 2015-05-15 timolang@gmail.com 9a16865 Added doxygen comments to filesyste, checkbox, numupdown and
00012  * screen module. And some minor changes to the other modules.
00013  *****/
00014
00015 #include "filesystem.h"
00016 #include "ll_filesystem.h"
00017
00018 bool filesystem_init()
00019 {
00020     return ll_filesystem_init();
00021 }
00022
00023 DIRECTORY_STRUCT* filesystem_dir_open(const char* path)
00024 {
00025     return ll_filesystem_dir_open(path);
00026 }
00027
00028 void filesystem_dir_close(DIRECTORY_STRUCT* dir)
00029 {
00030     ll_filesystem_dir_close(dir);
00031 }
00032
00033 FILE_HANDLE* filesystem_file_open(const char* filename)
00034 {
00035     return ll_filesystem_file_open(filename);
00036 }
00037
00038 void filesystem_file_close(FILE_HANDLE* handle)
00039 {
00040     ll_filesystem_file_close(handle);
00041 }
00042
00043 FILE_STATUS filesystem_file_seek(FILE_HANDLE* handle, uint32_t
offset)
00044 {
00045     return ll_filesystem_file_seek(handle, offset);
00046 }
00047
00048 FILE_STATUS filesystem_file_read(FILE_HANDLE* handle, uint8_t*
buf, uint32_t size)
00049 {
00050     return ll_filesystem_file_read(handle, buf, size);
00051 }
00052
00053 FILE_STATUS filesystem_file_write(FILE_HANDLE* handle, uint8_t*
buf, uint32_t size)
00054 {
00055     return ll_filesystem_file_write(handle, buf, size);
00056 }

```

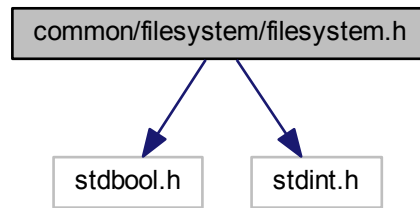
7.43 common/filesystem/filesystem.h File Reference

```

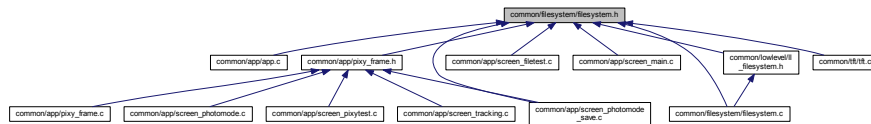
#include <stdbool.h>
#include <stdint.h>

```

Include dependency graph for filesystem.h:



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



Data Structures

- struct [FILE_DATE_STRUCT](#)
- struct [FILE_TIME_STRUCT](#)
- struct [FILE_STRUCT](#)
- struct [DIRECTORY_STRUCT](#)
- struct [FILE_HANDLE](#)

Enumerations

- enum [FILE_ATTRIBUTES](#) {
[F_RDO](#) = 0x01, [F_HID](#) = 0x02, [F_SYS](#) = 0x04, [F_DIR](#) = 0x10,
[F_ARC](#) = 0x20 }
- enum [FILE_STATUS](#) {
[F_OK](#), [F_EOF](#), [F_EACCESS](#), [F_INVALIDPARAM](#),
[F_DISKERROR](#) }

Functions

- bool [filesystem_init](#) ()
- [DIRECTORY_STRUCT](#) * [filesystem_dir_open](#) (const char *path)
- void [filesystem_dir_close](#) ([DIRECTORY_STRUCT](#) *dir)
- [FILE_HANDLE](#) * [filesystem_file_open](#) (const char *filename)
- void [filesystem_file_close](#) ([FILE_HANDLE](#) *handle)
- [FILE_STATUS](#) [filesystem_file_seek](#) ([FILE_HANDLE](#) *handle, uint32_t offset)
- [FILE_STATUS](#) [filesystem_file_read](#) ([FILE_HANDLE](#) *handle, uint8_t *buf, uint32_t size)
- [FILE_STATUS](#) [filesystem_file_write](#) ([FILE_HANDLE](#) *handle, uint8_t *buf, uint32_t size)

7.44 filesystem.h

```

00001 /*****
00002  * Project:      discoverpixy
00003  * Website:     https://github.com/t-moe/discoverpixy
00004  * Authors:     Aaron Schmocker, Timo Lang
00005  * Institution: BFH Bern University of Applied Sciences
00006  * File:       common/filesystem/filesystem.h
00007  *
00008  * Version History:
00009  * Date      Autor Email      SHA      Changes
00010  * 2015-04-03 timolang@gmail.com 51089aa Refactored Project Structure for use with emulator
00011  * 2015-05-10 timolang@gmail.com e2bce8f Added filesystem module, tests and implementation for it in
00012  * 2015-05-15 timolang@gmail.com 9a16865 Added doxygen comments to filesyste, checkbox, numupdown and
00013  * screen module. And some minor changes to the other modules.
00014  *****/
00015
00016 #ifndef FILESYSTEM_H
00017 #define FILESYSTEM_H
00018
00019 #include <stdbool.h>
00020 #include <stdint.h>
00021
00022
00023 typedef enum {
00024     F_RDO = 0x01,
00025     F_HID = 0x02,
00026     F_SYS = 0x04,
00027     F_DIR = 0x10,
00028     F_ARC = 0x20
00029 } FILE_ATTRIBUTES;
00030
00031 typedef struct {
00032     unsigned year : 7;
00033     unsigned month: 4;
00034     unsigned day: 5;
00035 } FILE_DATE_STRUCT;
00036
00037 typedef struct {
00038     unsigned hour : 5;
00039     unsigned min: 6;
00040     unsigned sec: 5;
00041 } FILE_TIME_STRUCT;
00042
00043 typedef struct {
00044     uint32_t fsize;
00045     FILE_DATE_STRUCT fdate;
00046     FILE_TIME_STRUCT ftime;
00047     uint8_t fattrib;
00048     char* fname;
00049 } FILE_STRUCT;
00050
00051 typedef struct {
00052     const char* path;
00053     uint16_t num_files;
00054     FILE_STRUCT* files;
00055 } DIRECTORY_STRUCT;
00056
00057 typedef struct {
00058     const char* fname;
00059     uint32_t fpos;
00060     uint32_t fsize;
00061 } FILE_HANDLE;
00062
00063 typedef enum {
00064     F_OK,
00065     F_EOF,
00066     F_EACCESS,
00067     F_INVALIDPARAM,
00068     F_DISKERROR
00069 } FILE_STATUS;
00070
00071 bool filesystem_init();
00072
00073 DIRECTORY_STRUCT* filesystem_dir_open(const char* path);
00074
00075 void filesystem_dir_close(DIRECTORY_STRUCT* dir);
00076
00077 FILE_HANDLE* filesystem_file_open(const char* filename);
00078
00079 void filesystem_file_close(FILE_HANDLE* handle);
00080

```



```

00137 FILE_STATUS filesystem_file_seek(FILE_HANDLE* handle, uint32_t
      offset);
00138
00146 FILE_STATUS filesystem_file_read(FILE_HANDLE* handle, uint8_t*
      buf, uint32_t size);
00147
00156 FILE_STATUS filesystem_file_write(FILE_HANDLE* handle, uint8_t*
      buf, uint32_t size);
00157
00158
00161 #endif /* FILESYSTEM_H */

```

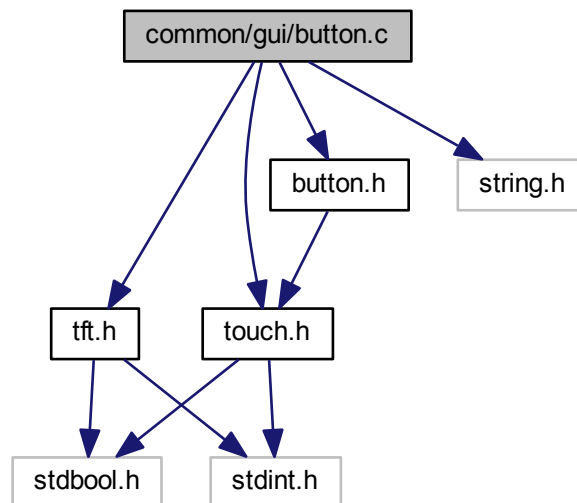
7.45 common/gui/button.c File Reference

```

#include "tft.h"
#include "touch.h"
#include "button.h"
#include <string.h>

```

Include dependency graph for button.c:



Macros

- `#define BRIGHTNESS_VAL 3`

Functions

- void `calculate_shadows` (uint16_t bgcolor, uint16_t *light_shadow, uint16_t *dark_shadow)
- void `buttons_cb` (void *touchArea, TOUCH_ACTION triggeredAction)
- bool `gui_button_add` (BUTTON_STRUCT *button)
- void `gui_button_redraw` (BUTTON_STRUCT *button)
- void `gui_button_remove` (BUTTON_STRUCT *button)

7.45.1 Macro Definition Documentation

7.45.1.1 #define BRIGHTNESS_VAL 3

7.45.2 Function Documentation

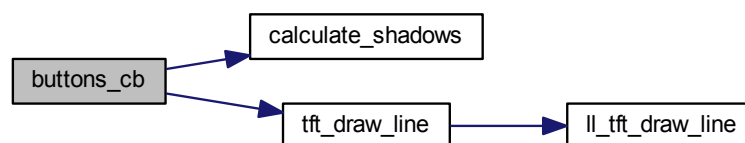
7.45.2.1 void buttons_cb(void * touchArea, TOUCH_ACTION triggeredAction)

Definition at line 92 of file [button.c](#).

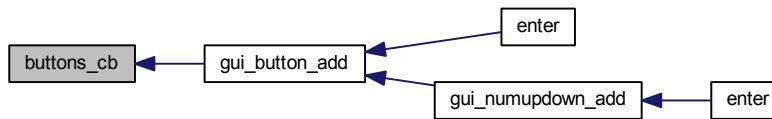
```

00093 {
00094     TOUCH_AREA_STRUCT* area = (TOUCH_AREA_STRUCT*)touchArea;
00095     BUTTON_STRUCT* button = (BUTTON_STRUCT*)touchArea;
00096
00097     uint16_t c_light, c_dark; //c_light and c_dark will be filled with a lighter and a darker color as the
                                background color (for the shadows)
00098     calculate_shadows(button->bgcolor, &c_light, &c_dark);
00099
00100     switch (triggeredAction) {
00101     case PEN_DOWN: //If the user touches the area for the "first time"
00102         area->hookedActions = PEN_UP | PEN_LEAVE; //for the future we only want
                                PEN_UP and PEN_LEAVE events
00103
00104         //Draw shadows
00105         tft_draw_line(button->base.x1 + 1, button->base.
00106 y1, button->base.x2 - 1, button->base.y1, c_dark); //North
00107         tft_draw_line(button->base.x1, button->base.y1 + 1, button->
00108 base.x1, button->base.y2 - 1, c_dark); //West
00109         tft_draw_line(button->base.x1 + 1, button->base.
00110 y2, button->base.x2 - 1, button->base.y2, c_light); //South
00111         tft_draw_line(button->base.x2, button->base.y1 + 1, button->
00112 base.x2, button->base.y2 - 1, c_light); //East
00113         break;
00114
00115     case PEN_UP: //If the user took the pen away, while in the area (=button pressed!)
00116     case PEN_LEAVE: //or the user "slided out" of the area
00117         area->hookedActions = PEN_DOWN; //for the future we only want PEN_DOWN events
00118
00119         //Draw inverse shadows
00120         tft_draw_line(button->base.x1 + 1, button->base.
00121 y1, button->base.x2 - 1, button->base.y1, c_light); //North
00122         tft_draw_line(button->base.x1, button->base.y1 + 1, button->
00123 base.x1, button->base.y2 - 1, c_light); //West
00124         tft_draw_line(button->base.x1 + 1, button->base.
00125 y2, button->base.x2 - 1, button->base.y2, c_dark); //South
00126         tft_draw_line(button->base.x2, button->base.y1 + 1, button->
00127 base.x2, button->base.y2 - 1, c_dark); //East
00128
00129         if (triggeredAction == PEN_UP && button->callback != NULL) { //If the button got
                                "pressed" instead of left, and the user provided a callback
00130             button->callback(button); //execute the user callback
00131         }
00132         break;
00133
00134     default:
00135         break;
00136     }
00137 }
00138 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



7.45.2.2 void calculate_shadows (uint16_t bgcolor, uint16_t * light_shadow, uint16_t * dark_shadow)

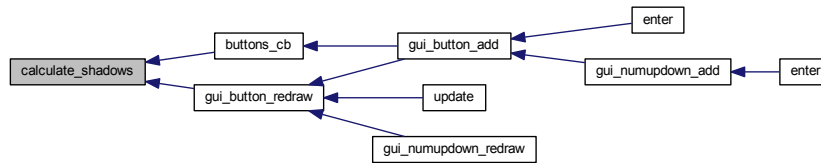
Definition at line 34 of file [button.c](#).

```

00035 {
00036 #define BRIGHTNESS_VAL 3 //How much the Brightness is in/decreased for button shadows (3 -> Add/Subtract 1/
3 off Full Value)
00037
00038     uint16_t c_light, c_dark; //c_light and c_dark will be filled with a lighter and a darker color as the
background color (for the shadows)
00039     uint8_t r, g, b;
00040
00041     //separate the channels of the 16-bit rgb565 color
00042     r = (bgcolor & 0xF800) >> 11;
00043     g = (bgcolor & 0x07E0) >> 5;
00044     b = (bgcolor & 0x001F) >> 0;
00045
00046     //For the light shadow color:
00047     if ((r + 0x1F / BRIGHTNESS_VAL) > 0x1F) { //Adding one third would exceed the maximum of
the red channel
00048         c_light = 0xF800; //Use full red
00049     } else { //adding one third to the red channel is fine
00050         c_light = (r + 0x1F / BRIGHTNESS_VAL) << 11; //Use same red as in the background,
but add one third
00051     }
00052
00053     if ((g + 0x3F / BRIGHTNESS_VAL) > 0x3F) { //same for the green channel
00054         c_light |= 0x07E0;
00055     } else {
00056         c_light |= (g + 0x3F / BRIGHTNESS_VAL) << 5;
00057     }
00058
00059     if ((b + 0x1F / BRIGHTNESS_VAL) > 0x1F) { //and the blue channel
00060         c_light |= 0x0018;
00061     } else {
00062         c_light |= (b + 0x1F / BRIGHTNESS_VAL) << 0;
00063     }
00064
00065     //For the dark shadow color
00066     if (r > (0x1F / BRIGHTNESS_VAL)) { //Subtracting one third would NOT exceed the minimum
of the red channel
00067         c_dark = (r - 0x1F / BRIGHTNESS_VAL) << 11; //Use same red as in the background,
but subtract one third
00068     } else { //Subtracting one third would give us a number below zero
00069         c_dark = 0x0000; //use no red channel
00070     }
00071
00072     if (g > (0x3F / BRIGHTNESS_VAL)) { //Same for the green channel
00073         c_dark |= (g - 0x3F / BRIGHTNESS_VAL) << 5;
00074     }
00075
00076     if (b > (0x1F / BRIGHTNESS_VAL)) { //and the blue channel
00077         c_dark |= (b - 0x1F / BRIGHTNESS_VAL) << 0;
00078     }
00079
00080     //Assign the calculated shadows to out parameters
00081     if (light_shadow != NULL) {
00082         *light_shadow = c_light;
00083     }
00084
00085     if (dark_shadow != NULL) {
00086         *dark_shadow = c_dark;
00087     }
00088
00089 }

```

Here is the caller graph for this function:



7.46 button.c

```

00001 /*****
00002  * Project:      discoverpixy
00003  * Website:     https://github.com/t-moe/discoverpixy
00004  * Authors:     Aaron Schmocker, Timo Lang
00005  * Institution:  BFH Bern University of Applied Sciences
00006  * File:        common/gui/button.c
00007  *
00008  * Version History:
00009  * Date          Autor Email      SHA      Changes
00010  * 2015-04-27    timolang@gmail.com 7c9eabc Added button support.
00011  * 2015-05-17    timolang@gmail.com 2d46336 Improved comments in implementation of button, checkbox,
00012  * numupdown, tft, touch and screen modules/submodules.
00013  *****/
00014
00015 #include "tft.h"
00016 #include "touch.h"
00017 #include "button.h"
00018 #include <string.h>
00019
00020 /* The Idea is as follows:
00021  * When the user add's a button we create a touch area for that region and wait for PEN_DOWN events.
00022  * Once the user puts the pen down in this area we'll redraw the button with different shadows (feedback)
00023  * and we'll now wait on PEN_UP or PEN_LEAVE events.
00024  * If the user takes the pen away while in the area (PEN_UP), we call the provided user callback
00025  * Otherwise (PEN_LEAVE) we only restore the initial shadows
00026  */
00027
00028 /* Possible improvements:
00029  * Move the button by 1 pixel while he is pressed, to create a "full 3d" experience
00030  * Add events for the case when the button is pressed for a long time, without release
00031  */
00032
00033 //Method to calculate the shadow colors used to create the "3d" effect
00034 void calculate_shadows(uint16_t bgcolor, uint16_t* light_shadow, uint16_t* dark_shadow)
00035 {
00036 #define BRIGHTNESS_VAL 3 //How much the Brightness is in/decreased for button shadows (3 -> Add/Subtract 1/
00037 3 off Full Value)
00038     uint16_t c_light, c_dark; //c_light and c_dark will be filled with a lighter and a darker color as the
00039     background color (for the shadows)
00040     uint8_t r, g, b;
00041
00042     //separate the channels of the 16-bit rgb565 color
00043     r = (bgcolor & 0xF800) >> 11;
00044     g = (bgcolor & 0x07E0) >> 5;
00045     b = (bgcolor & 0x001F) >> 0;
00046
00047     //For the light shadow color;
00048     if ((r + 0x1F / BRIGHTNESS_VAL) > 0x1F) { //Adding one third would exceed the maximum of
00049 the red channel
00050         c_light = 0xF800; //Use full red
00051     } else { //adding one third to the red channel is fine
00052         c_light = (r + 0x1F / BRIGHTNESS_VAL) << 11; //Use same red as in the background,
00053 but add one third
00054     }
00055
00056     if ((g + 0x3F / BRIGHTNESS_VAL) > 0x3F) { //same for the green channel
00057         c_light |= 0x07E0;
00058     } else {
00059         c_light |= (g + 0x3F / BRIGHTNESS_VAL) << 5;
00060     }
00061
00062     //For the dark shadow color;
00063     if ((r - 0x1F / BRIGHTNESS_VAL) < 0) { //Subtracting one third would go below 0
00064         c_dark = 0;
00065     } else {
00066         c_dark = (r - 0x1F / BRIGHTNESS_VAL) << 11;
00067     }
00068
00069     if ((g - 0x3F / BRIGHTNESS_VAL) < 0) {
00070         c_dark |= 0;
00071     } else {
00072         c_dark |= (g - 0x3F / BRIGHTNESS_VAL) << 5;
00073     }
00074
00075     if ((b - 0x1F / BRIGHTNESS_VAL) < 0) {
00076         c_dark = 0;
00077     } else {
00078         c_dark = (b - 0x1F / BRIGHTNESS_VAL) << 0;
00079     }
00080
00081     *light_shadow = c_light;
00082     *dark_shadow = c_dark;
00083 }
00084
00085 /* Button structure
00086  * struct button {
00087  *     uint16_t x, y, x2, y2; //Button coordinates
00088  *     uint16_t* light_shadow; //Light shadow color
00089  *     uint16_t* dark_shadow; //Dark shadow color
00090  *     void (*callback)(void); //User callback
00091  * };
00092  */
00093
00094 /* Button functions
00095  * void button_add(uint16_t x, uint16_t y, uint16_t x2, uint16_t y2, void (*callback)(void))
00096  * void button_redraw(uint16_t x, uint16_t y, uint16_t x2, uint16_t y2)
00097  * void button_remove(uint16_t x, uint16_t y, uint16_t x2, uint16_t y2)
00098  * void button_update(uint16_t x, uint16_t y, uint16_t x2, uint16_t y2)
00099  * void button_callback(void)
00100  */

```

```

00058
00059     if ((b + 0x1F / BRIGHTNESS_VAL) > 0x1F) { //and the blue channel
00060         c_light |= 0x0018;
00061     } else {
00062         c_light |= (b + 0x1F / BRIGHTNESS_VAL) << 0;
00063     }
00064
00065     //For the dark shadow color
00066     if (r > (0x1F / BRIGHTNESS_VAL)) { //Subtracting one third would NOT exceed the minimum
of the red channel
00067         c_dark = (r - 0x1F / BRIGHTNESS_VAL) << 11; //Use same red as in the background,
but subtract one third
00068     } else { //Subtracting one third would give us a number below zero
00069         c_dark = 0x0000; //use no red channel
00070     }
00071
00072     if (g > (0x3F / BRIGHTNESS_VAL)) { //Same for the green channel
00073         c_dark |= (g - 0x3F / BRIGHTNESS_VAL) << 5;
00074     }
00075
00076     if (b > (0x1F / BRIGHTNESS_VAL)) { //and the blue channel
00077         c_dark |= (b - 0x1F / BRIGHTNESS_VAL) << 0;
00078     }
00079
00080     //Assign the calculated shadows to out parameters
00081     if (light_shadow != NULL) {
00082         *light_shadow = c_light;
00083     }
00084
00085     if (dark_shadow != NULL) {
00086         *dark_shadow = c_dark;
00087     }
00088
00089 }
00090
00091 //Callback which is called when the user touches the touch-area we created for the button
00092 void buttons_cb(void* touchArea, TOUCH_ACTION triggeredAction)
00093 {
00094     TOUCH_AREA_STRUCT* area = (TOUCH_AREA_STRUCT*)touchArea;
00095     BUTTON_STRUCT* button = (BUTTON_STRUCT*)touchArea;
00096
00097     uint16_t c_light, c_dark; //c_light and c_dark will be filled with a lighter and a darker color as the
background color (for the shadows)
00098     calculate_shadows(button->bgcolor, &c_light, &c_dark);
00099
00100     switch (triggeredAction) {
00101     case PEN_DOWN: //If the user touches the area for the "first time"
00102         area->hookedActions = PEN_UP | PEN_LEAVE; //for the future we only want
PEN_UP and PEN_LEAVE events
00103
00104         //Draw shadows
00105         tft_draw_line(button->base.x1 + 1, button->base.
y1, button->base.x2 - 1, button->base.y1, c_dark); //North
00106         tft_draw_line(button->base.x1, button->base.y1 + 1, button->
base.x1, button->base.y2 - 1, c_dark); //West
00107         tft_draw_line(button->base.x1 + 1, button->base.
y2, button->base.x2 - 1, button->base.y2, c_light); //South
00108         tft_draw_line(button->base.x2, button->base.y1 + 1, button->
base.x2, button->base.y2 - 1, c_light); //East
00109         break;
00110
00111     case PEN_UP: //If the user took the pen away, while in the area (=button pressed!)
00112     case PEN_LEAVE: //or the user "slided out" of the area
00113         area->hookedActions = PEN_DOWN; //for the future we only want PEN_DOWN events
00114
00115         //Draw inverse shadows
00116         tft_draw_line(button->base.x1 + 1, button->base.
y1, button->base.x2 - 1, button->base.y1, c_light); //North
00117         tft_draw_line(button->base.x1, button->base.y1 + 1, button->
base.x1, button->base.y2 - 1, c_light); //West
00118         tft_draw_line(button->base.x1 + 1, button->base.
y2, button->base.x2 - 1, button->base.y2, c_dark); //South
00119         tft_draw_line(button->base.x2, button->base.y1 + 1, button->
base.x2, button->base.y2 - 1, c_dark); //East
00120
00121         if (triggeredAction == PEN_UP && button->callback != NULL) { //If the button got
"pressed" instead of left, and the user provided a callback
00122             button->callback(button); //execute the user callback
00123         }
00124
00125         break;
00126
00127     default:
00128         break;
00129     }
00130 }
00131

```

```

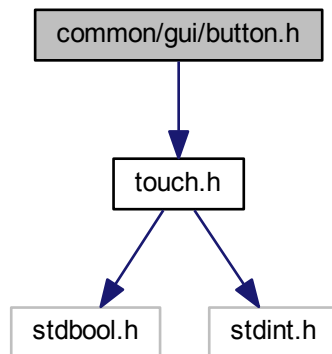
00132
00133 bool gui_button_add(BUTTON_STRUCT* button)
00134 {
00135     if (touch_have_empty(1)) { //Check if the touch module can handle one additional area
00136         //Calculate width and height of the button text
00137         unsigned int strwidth = tft_font_width(button->font) * strlen(button->
text);
00138         unsigned char strheight = tft_font_height(button->font);
00139
00140         button->base.hookedActions = PEN_DOWN; //At first we are interested in
PEN_DOWN events
00141         button->base.callback = buttons_cb; //Use our own callback for the touch area
events
00142
00143         if (button->base.x2 == AUTO) { //The user wants us to calculate the button width
automatically
00144             //Use string width + half of a character width as button width
00145             button->base.x2 = button->base.x1 - 1 + strwidth + (
tft_font_width(button->font) / 2);
00146         } else if ((button->base.x2 - button->base.x1 + 1) < (strwidth + 2)) { //the provided
width is too small to fit the entire text
00147             return false; //report error
00148         }
00149
00150         if (button->base.y2 == AUTO) { //The user wants us to calculate the button height
automatically
00151             //Use one and a half character heights as button height
00152             button->base.y2 = button->base.y1 - 1 + strheight + (strheight / 2);
00153         } else if ((button->base.y2 - button->base.y1 + 1) < (strheight + 2)) { //the provided
height is too small to fit the text
00154             return false;
00155         }
00156
00157         gui_button_redraw(button); //call the redraw method, which will take care of
drawing the entire button
00158         return touch_register_area(&button->base); //Register the touch area and
receive events for this button, from now on
00159     }
00160
00161     return false; //no more touch areas left
00162 }
00163
00164 void gui_button_redraw(BUTTON_STRUCT* button)
00165 {
00166     //Calculate text dimensions and shadow colors
00167     unsigned int strwidth = tft_font_width(button->font) * strlen(button->
text);
00168     unsigned char strheight = tft_font_height(button->font);
00169     uint16_t c_light, c_dark;
00170     calculate_shadows(button->bgcolor, &c_light, &c_dark);
00171
00172     //Draw the background and the 4 lines (shadow colors)
00173     tft_fill_rectangle(button->base.x1 + 1, button->base.
y1 + 1, button->base.x2 - 1, button->base.y2 - 1, button->bgcolor);
00174     tft_draw_line(button->base.x1 + 1, button->base.y1, button->
base.x2 - 1, button->base.y1, c_light); //North
00175     tft_draw_line(button->base.x1, button->base.y1 + 1, button->
base.x1, button->base.y2 - 1, c_light); //West
00176     tft_draw_line(button->base.x1 + 1, button->base.y2, button->
base.x2 - 1, button->base.y2, c_dark); //South
00177     tft_draw_line(button->base.x2, button->base.y1 + 1, button->
base.x2, button->base.y2 - 1, c_dark); //East
00178
00179     //Draw the text
00180     tft_print_line(button->base.x1 + (button->base.x2 - button->
base.x1 + 1 - strwidth) / 2, button->base.y1 + (button->base.y2 - button->
base.y1 + 1 - strheight) / 2, button->txtcolor, button->bgcolor, button->
font, button->text);
00181
00182 }
00183
00184 void gui_button_remove(BUTTON_STRUCT* button)
00185 {
00186     //We only need to unregister the touch area, as we have not allocated anything else
00187     touch_unregister_area((TOUCH_AREA_STRUCT*)button);
00188 }

```

7.47 common/gui/button.h File Reference

```
#include "touch.h"
```

Include dependency graph for button.h:



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



Data Structures

- struct [BUTTON_STRUCT](#)

Macros

- `#define` [AUTO](#) 0

Use this value instead of `x2`, `y2` in the [BUTTON_STRUCT](#) to autocalculate the button width/height.

Typedefs

- typedef void(* [BUTTON_CALLBACK](#)) (void *button)

Functions

- bool [gui_button_add](#) ([BUTTON_STRUCT](#) *button)
- void [gui_button_remove](#) ([BUTTON_STRUCT](#) *button)
- void [gui_button_redraw](#) ([BUTTON_STRUCT](#) *button)

7.48 button.h

```

00001 /*****
00002 * Project:      discoverpixy
00003 * Website:      https://github.com/t-moe/discoverpixy
00004 * Authors:      Aaron Schmocker, Timo Lang
00005 * Institution:   BFH Bern University of Applied Sciences
00006 * File:         common/gui/button.h
00007 *
00008 * Version History:
00009 * Date          Autor Email          SHA          Changes
00010 * 2015-04-27    timolang@gmail.com    7c9eabc    Added button support.
00011 * 2015-04-27    timolang@gmail.com    cf72baa    Introduced a Screen (sub) module and divided app into multiple
00012 * 2015-05-11    timolang@gmail.com    08d9fe0    More work on doxygen module structure
00013 * 2015-05-12    timolang@gmail.com    1402598    Added doxygen stuff for button module and some minor changes to
00014 * 2015-05-15    timolang@gmail.com    9a16865    Added doxygen comments to filesystem, checkbox, numupdown and
00015 * 2015-05-17    timolang@gmail.com    2d46336    Improved comments in implementation of button, checkbox,
00016 *              numupdown, tft, touch and screen modules/submodules.
00017 *****/
00018
00019 #ifndef BUTTON_H
00020 #define BUTTON_H
00021
00022 #include "touch.h"
00023
00029
00037
00042
00043
00049 typedef void (*BUTTON_CALLBACK)(void* button);
00050
00051
00055 typedef struct {
00056     TOUCH_AREA_STRUCT base;
00057     uint16_t bgcolor;
00058     BUTTON_CALLBACK callback;
00059     uint16_t txtcolor;
00060     uint8_t font;
00061     const char* text;
00062 } BUTTON_STRUCT;
00063
00064
00065 #define AUTO 0
00066
00067
00072 bool gui_button_add(BUTTON_STRUCT* button);
00073
00078 void gui_button_remove(BUTTON_STRUCT* button);
00079
00084 void gui_button_redraw(BUTTON_STRUCT* button);
00085
00086
00089 #endif /* BUTTON_H */

```

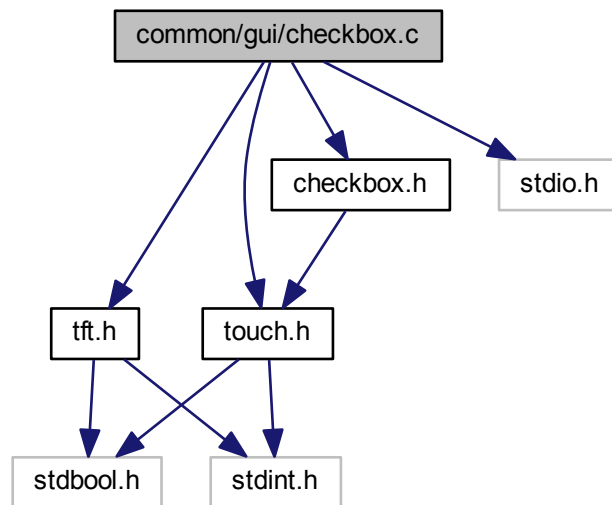
7.49 common/gui/checkbox.c File Reference

```

#include "tft.h"
#include "touch.h"
#include "checkbox.h"
#include <stdio.h>

```


Include dependency graph for checkbox.c:



Macros

- `#define ACTIVE_COLOR RGB(251,208,123)`
- `#define BORDER_COLOR RGB(29,82,129)`
- `#define BACKGROUND_COLOR WHITE`

Functions

- void `checkboxes_cb` (void *touchArea, `TOUCH_ACTION` triggeredAction)
- bool `gui_checkbox_add` (`CHECKBOX_STRUCT` *checkbox)
- void `gui_checkbox_redraw` (`CHECKBOX_STRUCT` *checkbox)
- void `gui_checkbox_remove` (`CHECKBOX_STRUCT` *checkbox)
- void `gui_checkbox_update` (`CHECKBOX_STRUCT` *checkbox)

7.49.1 Macro Definition Documentation

7.49.1.1 `#define ACTIVE_COLOR RGB(251,208,123)`

Definition at line 29 of file `checkbox.c`.

7.49.1.2 `#define BACKGROUND_COLOR WHITE`

Definition at line 31 of file `checkbox.c`.

7.49.1.3 `#define BORDER_COLOR RGB(29,82,129)`

Definition at line 30 of file `checkbox.c`.

7.49.2 Function Documentation

7.49.2.1 void checkboxes_cb (void * touchArea, TOUCH_ACTION triggeredAction)

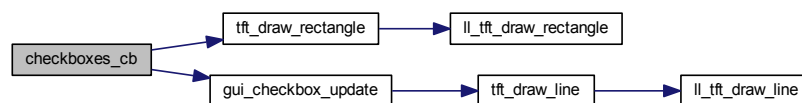
Definition at line 34 of file [checkbox.c](#).

```

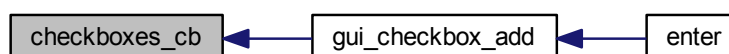
00035 {
00036     TOUCH_AREA_STRUCT* area = (TOUCH_AREA_STRUCT*)touchArea;
00037     CHECKBOX_STRUCT* checkbox = (CHECKBOX_STRUCT*)touchArea;
00038
00039     switch (triggeredAction) {
00040     case PEN_DOWN: //If the user touches the area for the "first time"
00041         area->hookedActions = PEN_UP | PEN_LEAVE; //for the future we only want
        PEN_UP and PEN_LEAVE events
00042
00043         //Draw active shadows
00044         tft_draw_rectangle(checkbox->base.x1 + 1, checkbox->
        base.y1 + 1, checkbox->base.x2 - 1, checkbox->base.y2 - 1,
        ACTIVE_COLOR);
00045         tft_draw_rectangle(checkbox->base.x1 + 2, checkbox->
        base.y1 + 2, checkbox->base.x2 - 2, checkbox->base.y2 - 2,
        ACTIVE_COLOR);
00046         break;
00047
00048     case PEN_UP: //If the user took the pen away, while in the area (=toggle checkbox!)
00049         checkbox->checked = !checkbox->checked; //Toggle checkbox state
00050         gui_checkbox_update(checkbox); //redraw/overdraw tickmark
00051
00052         if (checkbox->callback != NULL) { //The user provided a callback
00053             checkbox->callback(checkbox, checkbox->checked); //Call the provided callback
        with the new checked state
00054         }
00055
00056         // no break statement here!
00057     case PEN_LEAVE: //if the user "slided out" of the area
00058         area->hookedActions = PEN_DOWN; //for the future we only want PEN_DOWN events
00059
00060         //Draw inactive shadows
00061         tft_draw_rectangle(checkbox->base.x1 + 1, checkbox->
        base.y1 + 1, checkbox->base.x2 - 1, checkbox->base.y2 - 1,
        BACKGROUND_COLOR);
00062         tft_draw_rectangle(checkbox->base.x1 + 2, checkbox->
        base.y1 + 2, checkbox->base.x2 - 2, checkbox->base.y2 - 2,
        BACKGROUND_COLOR);
00063         break;
00064
00065     default:
00066         break;
00067     }
00068 }

```

Here is the call graph for this function:



Here is the caller graph for this function:



7.50 checkbox.c

```

00001 /*****
00002 * Project:      discoverpixy
00003 * Website:     https://github.com/t-moe/discoverpixy
00004 * Authors:     Aaron Schmocker, Timo Lang
00005 * Institution: BFH Bern University of Applied Sciences
00006 * File:       common/gui/checkbox.c
00007 *
00008 * Version History:
00009 * Date      Autor Email      SHA      Changes
00010 * 2015-04-27 timolang@gmail.com b300ac5 Added Checkbox support
00011 * 2015-05-17 timolang@gmail.com 2d46336 Improved comments in implementation of button, checkbox,
00012 * numupdown, tft, touch and screen modules/submodules.
00013 *****/
00014
00015 #include "tft.h"
00016 #include "touch.h"
00017 #include "checkbox.h"
00018 #include <stdio.h>
00019
00020 /* The idea is as follows:
00021 * When the user creates a checkbox we create a touch area for that region and wait for PEN_DOWN events.
00022 * Once the user puts the pen down in this area we'll redraw the checkbox with different shadows (feedback)
00023 * and we'll now wait on PEN_UP or PEN_LEAVE events.
00024 * If the user takes the pen away while in the area (PEN_UP), we toggle the checkbox and we call the
00025 * provided user callback
00026 * Otherwise (PEN_LEAVE) we only restore the initial shadows
00027 */
00028
00029 #define ACTIVE_COLOR RGB(251,208,123) //shadow color (inside of border)
00030 #define BORDER_COLOR RGB(29,82,129) //lpx border color
00031 #define BACKGROUND_COLOR WHITE //Background color
00032
00033 //Callback which is called when the user touches the touch-area we created for the checkbox
00034 void checkboxes_cb(void* touchArea, TOUCH_ACTION triggeredAction)
00035 {
00036     TOUCH_AREA_STRUCT* area = (TOUCH_AREA_STRUCT*)touchArea;
00037     CHECKBOX_STRUCT* checkbox = (CHECKBOX_STRUCT*)touchArea;
00038
00039     switch (triggeredAction) {
00040     case PEN_DOWN: //If the user touches the area for the "first time"
00041         area->hookedActions = PEN_UP | PEN_LEAVE; //for the future we only want
00042         PEN_UP and PEN_LEAVE events
00043
00044         //Draw active shadows
00045         tft_draw_rectangle(checkbox->base.x1 + 1, checkbox->
00046         base.y1 + 1, checkbox->base.x2 - 1, checkbox->base.y2 - 1,
00047         ACTIVE_COLOR);
00048         tft_draw_rectangle(checkbox->base.x1 + 2, checkbox->
00049         base.y1 + 2, checkbox->base.x2 - 2, checkbox->base.y2 - 2,
00050         ACTIVE_COLOR);
00051         break;
00052
00053     case PEN_UP: //If the user took the pen away, while in the area (=toggle checkbox!)
00054         checkbox->checked = !checkbox->checked; //Toggle checkbox state
00055         gui_checkbox_update(checkbox); //redraw/overdraw tickmark
00056
00057         if (checkbox->callback != NULL) { //The user provided a callback
00058             checkbox->callback(checkbox, checkbox->checked); //Call the provided callback
00059             with the new checked state
00060         }
00061
00062         // no break statement here!
00063     case PEN_LEAVE: //if the user "slided out" of the area
00064         area->hookedActions = PEN_DOWN; //for the future we only want PEN_DOWN events
00065
00066         //Draw inactive shadows
00067         tft_draw_rectangle(checkbox->base.x1 + 1, checkbox->
00068         base.y1 + 1, checkbox->base.x2 - 1, checkbox->base.y2 - 1,
00069         BACKGROUND_COLOR);
00070         tft_draw_rectangle(checkbox->base.x1 + 2, checkbox->
00071         base.y1 + 2, checkbox->base.x2 - 2, checkbox->base.y2 - 2,
00072         BACKGROUND_COLOR);
00073         break;
00074
00075     default:
00076         break;
00077     }
00078 }
00079
00080 bool gui_checkbox_add(CHECKBOX_STRUCT* checkbox)

```

```

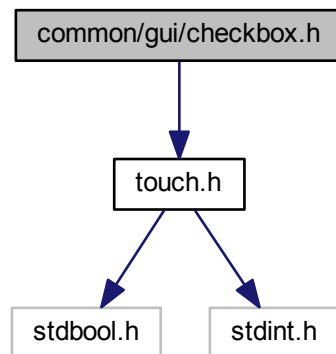
00071 {
00072     if (touch_have_empty(1)) { //Check if the touch module can handle one additional area
00073         unsigned char size = 0;
00074         checkbox->base.hookedActions = PEN_DOWN; //At first we are interested in
PEN_DOWN events
00075         checkbox->base.callback = checkboxes_cb; //Use our own callback for the
touch area events
00076
00077         //Check the size of the checkbox
00078         if (checkbox->base.x2 > checkbox->base.x1) {
00079             size = checkbox->base.x2 - checkbox->base.x1; //use width as size
00080         }
00081
00082         if (checkbox->base.y2 > checkbox->base.y1) {
00083             if ((checkbox->base.y2 - checkbox->base.y1) > size) { //height is larger than size
00084                 size = checkbox->base.y2 - checkbox->base.y1; //use height as size
00085             }
00086         }
00087
00088         if (size == 0) { //no size found (maybe swap x2 and x1 or y2 and y1 ?)
00089             return false; //signal error
00090         }
00091
00092         if ((size & 0x01)) { //the size is an odd number
00093             size++; //make size an even number
00094         }
00095
00096         //Correct x2,y2 so that the checkbox is quadratic
00097         checkbox->base.x2 = checkbox->base.x1 + size;
00098         checkbox->base.y2 = checkbox->base.y1 + size;
00099
00100         gui_checkbox_redraw(checkbox); //Call redraw method, which will take care of the
drawing of the entire checkbox
00101
00102         return touch_register_area(&checkbox->base); //Register the touch area and
receive events for this checkbox, from now on
00103     }
00104
00105     return false; //no more touch areas left
00106 }
00107
00108 void gui_checkbox_redraw(CHECKBOX_STRUCT* checkbox)
00109 {
00110     //Draw background and border
00111     tft_fill_rectangle(checkbox->base.x1 + 1, checkbox->
base.y1 + 1, checkbox->base.x2 - 1, checkbox->base.y2 - 1,
BACKGROUND_COLOR);
00112     tft_draw_rectangle(checkbox->base.x1, checkbox->base.
y1, checkbox->base.x2, checkbox->base.y2, BORDER_COLOR);
00113
00114     if (checkbox->checked) { //checkbox is currently checked
00115         gui_checkbox_update(checkbox); //Call update method which will draw the tickmark
00116     }
00117 }
00118
00119 void gui_checkbox_remove(CHECKBOX_STRUCT* checkbox)
00120 {
00121     //We only need to unregister the touch area, as we have not allocated anything else
00122     touch_unregister_area((TOUCH_AREA_STRUCT*)checkbox);
00123 }
00124
00125 void gui_checkbox_update(CHECKBOX_STRUCT* checkbox)
00126 {
00127     unsigned int c = (checkbox->checked) ? checkbox->fgcolor :
BACKGROUND_COLOR; //color to use for the tickmark
00128
00129     //helper points inside the checkbox
00130     unsigned int xcent = checkbox->base.x1 + (checkbox->base.x2 - checkbox->
base.x1) * 6 / 14;
00131     unsigned int yleft = checkbox->base.y2 - (xcent - checkbox->base.
x1) - 1;
00132     unsigned int yright = checkbox->base.y2 - (checkbox->base.x2 - xcent) - 1;
00133     unsigned int ybot = checkbox->base.y2 - 4;
00134
00135     //Draw tickmark as a 3pixel wide line
00136     tft_draw_line(checkbox->base.x1 + 3, yleft - 1, xcent, ybot - 1, c);
00137     tft_draw_line(checkbox->base.x1 + 3, yleft, xcent, ybot, c);
00138     tft_draw_line(checkbox->base.x1 + 3, yleft + 1, xcent, ybot + 1, c);
00139     xcent++;
00140     ybot--;
00141     tft_draw_line(xcent, ybot - 1, checkbox->base.x2 - 3, yright - 1, c);
00142     tft_draw_line(xcent, ybot, checkbox->base.x2 - 3, yright + 0, c);
00143     tft_draw_line(xcent, ybot + 1, checkbox->base.x2 - 3, yright + 1, c);
00144 }

```

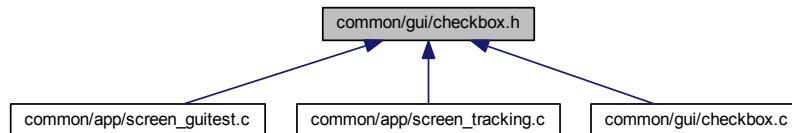
7.51 common/gui/checkbox.h File Reference

```
#include "touch.h"
```

Include dependency graph for checkbox.h:



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



Data Structures

- struct [CHECKBOX_STRUCT](#)

Macros

- `#define` [CHECKBOX_WIN_FG_COLOR](#) [RGB](#)(32,161,34)

Typedefs

- typedef void(* [CHECKBOX_CALLBACK](#)) (void *checkbox, bool checked)

Functions

- bool [gui_checkbox_add](#) ([CHECKBOX_STRUCT](#) *checkbox)
- void [gui_checkbox_remove](#) ([CHECKBOX_STRUCT](#) *checkbox)
- void [gui_checkbox_update](#) ([CHECKBOX_STRUCT](#) *checkbox)
- void [gui_checkbox_redraw](#) ([CHECKBOX_STRUCT](#) *checkbox)

7.52 checkbox.h

```

00001 /*****
00002  * Project:      discoverpixy
00003  * Website:     https://github.com/t-moe/discoverpixy
00004  * Authors:     Aaron Schmocker, Timo Lang
00005  * Institution: BFH Bern University of Applied Sciences
00006  * File:       common/gui/checkbox.h
00007  *
00008  * Version History:
00009  * Date      Autor Email      SHA      Changes
00010  * 2015-04-27 timolang@gmail.com b300ac5 Added Checkbox support
00011  * 2015-05-11 timolang@gmail.com 08d9fe0 More work on doxygen module structure
00012  * 2015-05-15 timolang@gmail.com 9a16865 Added doxygen comments to filesystem, checkbox, numupdown and
00013  * 2015-05-17 timolang@gmail.com 2d46336 Improved comments in implementation of button, checkbox,
00014  * numupdown, tft, touch and screen modules/submodules.
00015  *****/
00016
00017 #ifndef CHECKBOX_H
00018 #define CHECKBOX_H
00019
00020 #include "touch.h"
00021
00022
00023
00024
00025
00045 typedef void (*CHECKBOX_CALLBACK)(void* checkbox, bool checked);
00046
00050 typedef struct {
00051     TOUCH_AREA_STRUCT base;
00052     uint16_t fgcolor;
00053     bool checked;
00054     CHECKBOX_CALLBACK callback;
00055 } CHECKBOX_STRUCT;
00056
00062 bool gui_checkbox_add(CHECKBOX_STRUCT* checkbox);
00063
00068 void gui_checkbox_remove(CHECKBOX_STRUCT* checkbox);
00069
00074 void gui_checkbox_update(CHECKBOX_STRUCT* checkbox);
00075
00080 void gui_checkbox_redraw(CHECKBOX_STRUCT* checkbox);
00081
00082 #define CHECKBOX_WIN_FG_COLOR RGB(32,161,34)
00083
00086 #endif /* CHECKBOX_H */

```

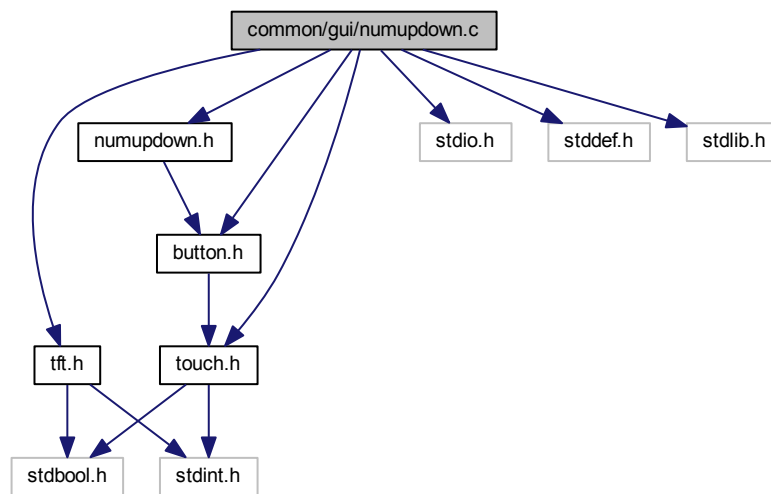
7.53 common/gui/numupdown.c File Reference

```

#include "tft.h"
#include "touch.h"
#include "button.h"
#include "numupdown.h"
#include <stdio.h>
#include <stddef.h>
#include <stdlib.h>

```

Include dependency graph for numupdown.c:



Macros

- `#define BASE_COLOR RGB(90,90,90)`

Functions

- void `button_up_cb` (void *button)
- void `button_down_cb` (void *button)
- static uint8_t `calc_text_width` (int16_t val)
- bool `gui_numupdown_add` (NUMUPDOWN_STRUCT *numupdown)
- void `gui_numupdown_remove` (NUMUPDOWN_STRUCT *numupdown)
- void `gui_numupdown_redraw` (NUMUPDOWN_STRUCT *numupdown)
- void `gui_numupdown_update` (NUMUPDOWN_STRUCT *numupdown)

7.53.1 Macro Definition Documentation

7.53.1.1 `#define BASE_COLOR RGB(90,90,90)`

Definition at line 31 of file `numupdown.c`.

7.53.2 Function Documentation

7.53.2.1 void `button_down_cb` (void * *button*)

Definition at line 50 of file `numupdown.c`.

```

00051 {
00052     //Get the pointer to the numupdown: subtract the offset of the buttonDown member in the struct from the
        button pointer
00053     NUMUPDOWN_STRUCT* element = button - offsetof(
        NUMUPDOWN_STRUCT, buttonDown);
00054

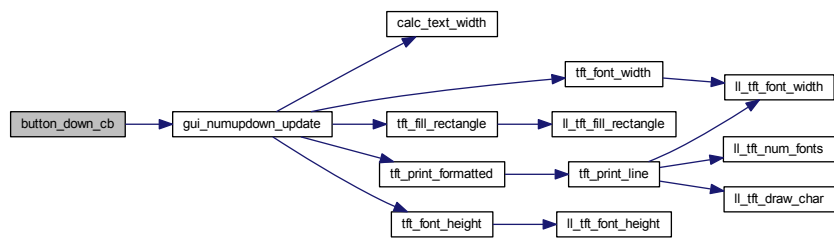
```

```

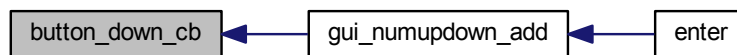
00055     if (element->value > element->min) { //old value lies above the minimum
00056         element->value--; //let's decrease the value
00057         gui_numupdown_update(element); //and redraw everything
00058
00059         if (element->callback != NULL) { //the user provided a callback
00060             element->callback(element, element->value); //Call the user callback with the new
00061             value
00062         }
00063     }

```

Here is the call graph for this function:



Here is the caller graph for this function:



7.53.2.2 void button_up_cb (void * button)

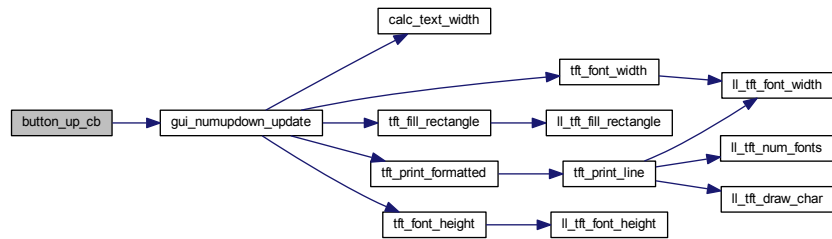
Definition at line 34 of file [numupdown.c](#).

```

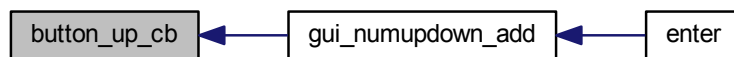
00035 {
00036     //Get the pointer to the numupdown: subtract the offset of the buttonUp member in the struct from the
00037     button pointer
00038     NUMUPDOWN_STRUCT* element = button - offsetof(
00039     NUMUPDOWN_STRUCT, buttonUp);
00040
00041     if (element->value < element->max) { //old value lies below the maximum
00042         element->value++; //let's increase the value
00043         gui_numupdown_update(element); //and redraw everything
00044
00045         if (element->callback != NULL) { //the user provided a callback
00046             element->callback(element, element->value); //Call the user callback with the new
00047             value
00048         }
00049     }

```


Here is the call graph for this function:



Here is the caller graph for this function:



7.53.2.3 static uint8_t calc_text_width (int16_t val) [static]

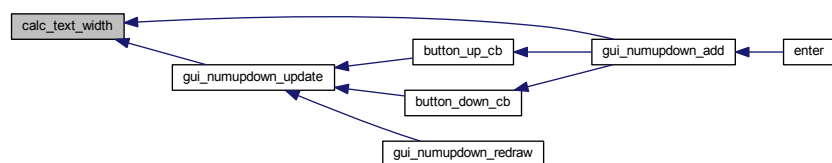
Definition at line 66 of file [numupdown.c](#).

```

00067 {
00068     uint8_t width = 1 + (val < 0); //1 if positive, 2 if negative (to let space for sign)
00069     val = abs(val); //Make the number positive
00070
00071     while (val >= 10) { //while we have two or more digits
00072         val /= 10; //remove one digit
00073         width++; //add one character
00074     }
00075
00076     return width;
00077 }

```

Here is the caller graph for this function:



7.54 numupdown.c

```
00001 /*****
```

```

*****
00002 * Project:      discoverpixy
00003 * Website:     https://github.com/t-moe/discoverpixy
00004 * Authors:     Aaron Schmocker, Timo Lang
00005 * Institution:  BFH Bern University of Applied Sciences
00006 * File:        common/gui/numupdown.c
00007 *
00008 * Version History:
00009 * Date          Autor Email          SHA          Changes
00010 * 2015-04-30    timolang@gmail.com    76ea9d8    Added num up down support.
00011 * 2015-04-30    timolang@gmail.com    b491b78    Made numupdown horizontal
00012 * 2015-05-15    timolang@gmail.com    9a16865    Added doxygen comments to filesyste, checkbox, numupdown and
               screen module. And some minor changes to the other modules.
00013 * 2015-05-17    timolang@gmail.com    2d46336    Improved comments in implementation of button, checkbox,
               numupdown, tft, touch and screen modules/submodules.
00014 *
00015 *****
00016
00017 #include "tft.h"
00018 #include "touch.h"
00019 #include "button.h"
00020 #include "numupdown.h"
00021 #include <stdio.h> //for sprintf
00022 #include <stddef.h> //for offsetof macro
00023 #include <stdlib.h> //for abs
00024
00025 /* The idea is as follows:
00026  * When the user add's a numupdown we create two buttons, one with a plus and one with a minus sign in it
00027  * When the user presses one of the buttons we check and increase the value and execute the provided user
               callback
00028  */
00029
00030
00031 #define BASE_COLOR RGB(90,90,90) //Background color for the whole element
00032
00033 //Callback which is called when the user presses the "plus" button
00034 void button_up_cb(void* button)
00035 {
00036     //Get the pointer to the numupdown: subtract the offset of the buttonUp member in the struct from the
               button pointer
00037     NUMUPDOWN_STRUCT* element = button - offsetof(
               NUMUPDOWN_STRUCT, buttonUp);
00038
00039     if (element->value < element->max) { //old value lies below the maximum
00040         element->value++; //let's increase the value
00041         gui_numupdown_update(element); //and redraw everything
00042
00043         if (element->callback != NULL) { //the user provided a callback
00044             element->callback(element, element->value); //Call the user callback with the new
               value
00045         }
00046     }
00047 }
00048
00049 //Callback which is called when the user presses the "minus" button
00050 void button_down_cb(void* button)
00051 {
00052     //Get the pointer to the numupdown: subtract the offset of the buttonDown member in the struct from the
               button pointer
00053     NUMUPDOWN_STRUCT* element = button - offsetof(
               NUMUPDOWN_STRUCT, buttonDown);
00054
00055     if (element->value > element->min) { //old value lies above the minimum
00056         element->value--; //let's decrease the value
00057         gui_numupdown_update(element); //and redraw everything
00058
00059         if (element->callback != NULL) { //the user provided a callback
00060             element->callback(element, element->value); //Call the user callback with the new
               value
00061         }
00062     }
00063 }
00064
00065 //Method to calculate the number of characters needed to print the provided number in decimal notation
               (with optional sign)
00066 static uint8_t calc_text_width(int16_t val)
00067 {
00068     uint8_t width = 1 + (val < 0); //1 if positive, 2 if negative (to let space for sign)
00069     val = abs(val); //Make the number positive
00070
00071     while (val >= 10) { //while we have two or more digits
00072         val /= 10; //remove one digit
00073         width++; //add one character
00074     }
00075
00076     return width;

```

```

00077 }
00078
00079
00080 bool gui_numupdown_add(NUMUPDOWN_STRUCT* numupdown)
00081 {
00082     if (touch_have_empty(2)) { //Check if the touch module can handle two additional areas
00083         if (numupdown->min > numupdown->max) { //min is bigger than max?
00084             return false; //invalid parameter
00085         }
00086
00087         if (numupdown->value < numupdown->min) { //value is smaller than min?
00088             numupdown->value = numupdown->min; //normalize value
00089         } else if (numupdown->value > numupdown->max) { //value is bigger than max?
00090             numupdown->value = numupdown->max; //normalize value
00091         }
00092
00093         uint8_t tw1 = calc_text_width(numupdown->max); //Calculate character width to
render maximum value
00094         uint8_t tw2 = calc_text_width(numupdown->min); //Calculate character width to
render minimum value
00095
00096         if (tw2 > tw1) {
00097             tw1 = tw2; //ensure tw1 contains the larger number of the two
00098         }
00099
00100         uint8_t width = tft_font_width(0) * (tw1 + 1); //Calculate width of the number area
00101
00102         //Add "minus" button to the left side of the number area
00103         numupdown->buttonDown.base.x1 = numupdown->x;
00104         numupdown->buttonDown.base.y1 = numupdown->y;
00105         numupdown->buttonDown.base.x2 = AUTO;
00106         numupdown->buttonDown.base.y2 = numupdown->y +
tft_font_height(0) * 2;
00107         numupdown->buttonDown.text = "-";
00108         numupdown->buttonDown.font = 0;
00109         numupdown->buttonDown.bgcolor = BASE_COLOR;
00110         numupdown->buttonDown.txtcolor = WHITE;
00111         numupdown->buttonDown.callback = button_down_cb;
00112         gui_button_add(&numupdown->buttonDown);
00113
00114         //Add "plus" button to the right side of the number area
00115         numupdown->buttonUp.base.x1 = numupdown->buttonDown.
base.x2
+ width + 2;
00116         numupdown->buttonUp.base.y1 = numupdown->y;
00117         numupdown->buttonUp.base.x2 = AUTO;
00118         numupdown->buttonUp.base.y2 = numupdown->y +
tft_font_height(0) * 2;
00119         numupdown->buttonUp.text = "+";
00120         numupdown->buttonUp.font = 0;
00121         numupdown->buttonUp.bgcolor = BASE_COLOR;
00122         numupdown->buttonUp.txtcolor = WHITE;
00123         numupdown->buttonUp.callback = button_up_cb;
00124         gui_button_add(&numupdown->buttonUp);
00125
00126         //Draw background and label of the number area
00127         tft_fill_rectangle(numupdown->buttonDown.
base.x2
+ 2, numupdown->y, numupdown->buttonDown.base.x2 + width, numupdown->
buttonUp.base.y2, BASE_COLOR);
00128         tft_print_formatted(numupdown->buttonDown.
base.x2
+ 2 + tft_font_width(0) / 2, numupdown->y +
tft_font_height(0) / 2, numupdown->fgcolor, BASE_COLOR, 0, "%*d", tw1,
numupdown->value);
00129
00130         return true;
00131     }
00132
00133     return false; //not enough touch areas left
00134 }
00135
00136 void gui_numupdown_remove(NUMUPDOWN_STRUCT* numupdown)
00137 {
00138     //remove the two buttons, we have no other allocated resources
00139     gui_button_remove(&numupdown->buttonUp);
00140     gui_button_remove(&numupdown->buttonDown);
00141 }
00142
00143
00144 void gui_numupdown_redraw(NUMUPDOWN_STRUCT* numupdown)
00145 {
00146     //redraw the two buttons
00147     gui_button_redraw(&numupdown->buttonUp);
00148     gui_button_redraw(&numupdown->buttonDown);
00149
00150     //call update method which will take care of the number-area rendering
00151     gui_numupdown_update(numupdown);
00152 }
00153

```

```

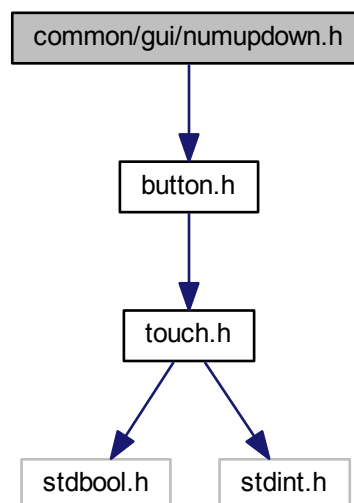
00154 void gui_numupdown_update(NUMUPDOWN_STRUCT* numupdown)
00155 {
00156     //Calculate the number area width again (see above)
00157     uint8_t tw1 = calc_text_width(numupdown->max);
00158     uint8_t tw2 = calc_text_width(numupdown->min);
00159
00160     if (tw2 > tw1) {
00161         tw1 = tw2;
00162     }
00163
00164     uint8_t width = tft_font_width(0) * (tw1 + 1);
00165
00166     //Draw background and label of the number area
00167     tft_fill_rectangle(numupdown->buttonDown.base.x2 + width, numupdown->
buttonUp.base.y2, BASE_COLOR);
00168     tft_print_formatted(numupdown->buttonDown.base.x2 + 2, numupdown->y + tft_font_height(0) / 2,
numupdown->fgcolor, BASE_COLOR, 0, "%*d", tw1, numupdown->value);
00169 }

```

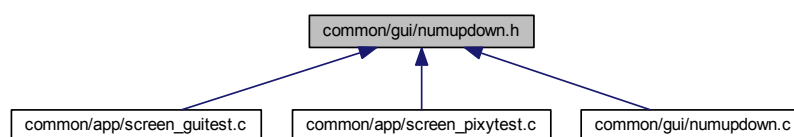
7.55 common/gui/numupdown.h File Reference

#include "button.h"

Include dependency graph for numupdown.h:



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



Data Structures

- struct [NUMUPDOWN_STRUCT](#)

Typedefs

- typedef void(* [NUMUPDOWN_CALLBACK](#)) (void *numupdown, int16_t value)

Functions

- bool [gui_numupdown_add](#) ([NUMUPDOWN_STRUCT](#) *numupdown)
- void [gui_numupdown_remove](#) ([NUMUPDOWN_STRUCT](#) *numupdown)
- void [gui_numupdown_update](#) ([NUMUPDOWN_STRUCT](#) *numupdown)
- void [gui_numupdown_redraw](#) ([NUMUPDOWN_STRUCT](#) *numupdown)

7.56 numupdown.h

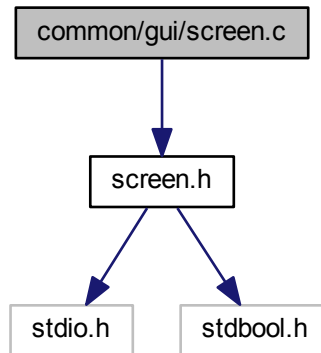
```

00001 /*****
00002 * Project:      discoverpixy
00003 * Website:      https://github.com/t-moe/discoverpixy
00004 * Authors:      Aaron Schmocker, Timo Lang
00005 * Institution:  BFH Bern University of Applied Sciences
00006 * File:         common/gui/numupdown.h
00007 *
00008 * Version History:
00009 * Date          Autor Email          SHA          Changes
00010 * 2015-04-30    timolang@gmail.com    76ea9d8    Added num up down support.
00011 * 2015-05-11    timolang@gmail.com    08d9fe0    More work on doxygen module structure
00012 * 2015-05-15    timolang@gmail.com    9a16865    Added doxygen comments to filesyste, checkbox, numupdown and
00013 *              screen module. And some minor changes to the other modules.
00014 *****/
00015
00016 #ifndef NUMUPDOWN_H
00017 #define NUMUPDOWN_H
00018
00019 #include "button.h"
00020
00025
00031
00036
00043 typedef void (*NUMUPDOWN_CALLBACK)(void* numupdown, int16_t value);
00044
00048 typedef struct {
00049     uint16_t x;
00050     uint16_t y;
00051     uint16_t fgcolor;
00052     int16_t value;
00053     int16_t min;
00054     int16_t max;
00055     NUMUPDOWN_CALLBACK callback;
00056
00057     BUTTON_STRUCT buttonUp;
00058     BUTTON_STRUCT buttonDown;
00059 } NUMUPDOWN_STRUCT;
00060
00066 bool gui_numupdown_add(NUMUPDOWN_STRUCT* numupdown);
00067
00072 void gui_numupdown_remove(NUMUPDOWN_STRUCT* numupdown);
00073
00078 void gui_numupdown_update(NUMUPDOWN_STRUCT* numupdown);
00079
00084 void gui_numupdown_redraw(NUMUPDOWN_STRUCT* numupdown);
00085
00088 #endif /* NUMUPDOWN_H */

```

7.57 common/gui/screen.c File Reference

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



Functions

- [SCREEN_STRUCT * gui_screen_get_current \(\)](#)
- void [gui_screen_update \(\)](#)
- bool [gui_screen_navigate \(SCREEN_STRUCT *screen\)](#)
- bool [gui_screen_back \(\)](#)

Variables

- static [SCREEN_STRUCT * screen_list = NULL](#)
- static [SCREEN_STRUCT * screen_current = NULL](#)
- static volatile [SCREEN_STRUCT * screen_goto = NULL](#)

7.57.1 Variable Documentation

7.57.1.1 **SCREEN_STRUCT*** `screen_current` = NULL [static]

Definition at line 32 of file [screen.c](#).

7.57.1.2 **volatile SCREEN_STRUCT*** `screen_goto` = NULL [static]

Definition at line 33 of file [screen.c](#).

7.57.1.3 **SCREEN_STRUCT*** `screen_list` = NULL [static]

Definition at line 31 of file [screen.c](#).

7.58 screen.c

```

00001 /*****
00002 * Project:      discoverpixy
00003 * Website:     https://github.com/t-moe/discoverpixy
00004 * Authors:     Aaron Schmocker, Timo Lang
00005 * Institution:  BFH Bern University of Applied Sciences
00006 * File:        common/gui/screen.c
00007 *
00008 * Version History:
00009 * Date      Autor Email      SHA      Changes
00010 * 2015-04-27 timolang@gmail.com cf72baa Introduced a Screen (sub) module and divided app into multiple
00011 *           screens.
00012 * 2015-04-27 timolang@gmail.com 77e6d0e Fixed screen implementation.
00013 * 2015-05-10 timolang@gmail.com b6ab7c8 Fixed compiler warning in tft and screen module.
00014 * 2015-05-17 timolang@gmail.com 2d46336 Improved comments in implementation of button, checkbox,
00015 *           numupdown, tft, touch and screen modules/submodules.
00016 * 2015-06-01 timolang@gmail.com eb573bc Finalized calibration. Fixed a bug in screen module.
00017 * 2015-06-06 timolang@gmail.com c06661d Fixed some outdated comments in source code. Documented Gui
00018 *           Module in docu.
00019 *****/
00020 #include "screen.h"
00021
00022 /* The idea is as follows:
00023 * We only call screen callbacks from the gui_screen_update() method, which is called from the applications
00024 * main loop.
00025 * Instructions to switch the screen will be delayed until the gui_screen_update() method is called again.
00026 * This makes it safe to change the screen from an touch interrupt (e.g. button callback)
00027 */
00028 /* Possible Improvements:
00029 * Ensure that you can not navigate to a screen which is already in the history (because it will corrupt
00030 * the list)
00031 */
00032 static SCREEN_STRUCT* screen_list = NULL; //Head of the linked list which stores
00033 the screen history.
00034 static SCREEN_STRUCT* screen_current = NULL; //Pointer to the current screen (=
00035 tail of the list)
00036 static volatile SCREEN_STRUCT* screen_goto = NULL; //Screen we should navigate to
00037 once we enter the gui_screen_update() method again
00038
00039 SCREEN_STRUCT* gui_screen_get_current()
00040 {
00041     return screen_current;
00042 }
00043
00044 void gui_screen_update()
00045 {
00046     if (screen_goto != NULL) { //we received the task to switch the screen
00047         SCREEN_STRUCT* go = (SCREEN_STRUCT*) screen_goto; //Backup volatile
00048         variable
00049         screen_goto = NULL; //reset the "goto instruction", since we're processing it now
00050
00051         if (go->next != NULL) { //The screen is not the last in the list, so we're going back
00052             if (go->next != screen_current) { //this condition should always be false
00053                 return; //list corrupted?
00054             }
00055
00056             screen_current->on_leave(screen_current); //let the current screen free/unregister it's
00057             resources
00058             go->next = NULL; //remove the current screen from the list
00059         } else { //we're going forward (to a new screen)
00060             if (screen_current != NULL) { //this is not the first screen
00061                 screen_current->on_leave(screen_current); //let the current screen free/unregister
00062                 it's resources
00063                 screen_current->next = go; //append the new screen to the end of the list
00064             } else { //first screen ever seen
00065                 screen_list = go; //set the new screen as list-head
00066             }
00067
00068             go->on_enter(go); //let the new screen allocate/register it's resources
00069             screen_current = go; //the new screen is now the current screen. Transition done
00070         }
00071
00072         if (screen_current != NULL) { //A screen has been set
00073             screen_current->on_update(screen_current); //Update current screen
00074         }
00075     }
00076 }

```


Data Structures

- struct [SCREEN_S](#)

Typedefs

- typedef void(* [SCREEN_CALLBACK](#)) (void *screen)
- typedef struct [SCREEN_S](#) [SCREEN_STRUCT](#)

Functions

- bool [gui_screen_navigate](#) ([SCREEN_STRUCT](#) *screen)
- bool [gui_screen_back](#) ()
- [SCREEN_STRUCT](#) * [gui_screen_get_current](#) ()
- void [gui_screen_update](#) ()

7.60 screen.h

```

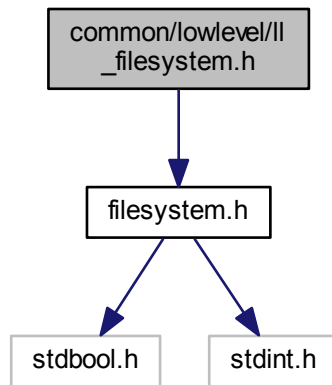
00001 /*****
00002  * Project:      discoverpixy
00003  * Website:     https://github.com/t-moe/discoverpixy
00004  * Authors:     Aaron Schmocker, Timo Lang
00005  * Institution:  BFH Bern University of Applied Sciences
00006  * File:        common/gui/screen.h
00007  *
00008  * Version History:
00009  * Date      Autor Email      SHA      Changes
00010  * 2015-04-27 timolang@gmail.com cf72baa Introduced a Screen (sub) module and divided app into multiple
00011  * 2015-04-27 timolang@gmail.com 77e6d0e Fixed screen implementation.
00012  * 2015-05-11 timolang@gmail.com 08d9fe0 More work on doxygen module structure
00013  * 2015-05-15 timolang@gmail.com 9a16865 Added doxygen comments to filesystem, checkbox, numupdown and
00014  * 2015-05-17 timolang@gmail.com 2d46336 Improved comments in implementation of button, checkbox,
00015  * 2015-06-06 timolang@gmail.com c06661d Fixed some outdated comments in source code. Documented Gui
00016  * Module in docu.
00017  *****/
00018
00019 #ifndef SCREEN_H
00020 #define SCREEN_H
00021
00022 #include <stdio.h>
00023 #include <stdbool.h>
00024
00025
00026
00027
00028
00029
00030
00031
00032
00033
00034
00035
00036
00037
00038
00039
00040
00041
00042
00043
00044
00045
00046
00047 typedef void (*SCREEN_CALLBACK)(void* screen);
00048
00049
00050
00051
00052 typedef struct SCREEN_S {
00053     SCREEN_CALLBACK on_enter;
00054     SCREEN_CALLBACK on_leave;
00055     SCREEN_CALLBACK on_update;
00056
00057     struct SCREEN_S* next;
00058 } SCREEN_STRUCT;
00059
00060
00061
00062
00063
00064
00065
00066
00067
00068
00069 bool gui_screen_navigate(SCREEN_STRUCT* screen);
00070
00071
00072
00073
00074
00075
00076 bool gui_screen_back();
00077
00078
00079
00080
00081
00082
00083 SCREEN_STRUCT* gui_screen_get_current();
00084
00085 //Updates/switches the screens. Call this from the app main loop, as fast as you can.
00086
00087
00088
00089 void gui_screen_update();
00090
00091
00092
00093
00094 #endif /* SCREEN_H */

```

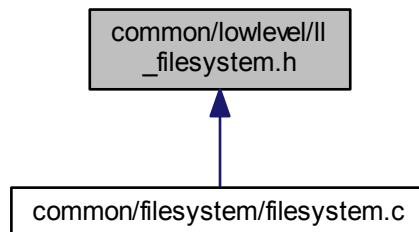
7.61 common/lowlevel/ll_filesystem.h File Reference

```
#include "filesystem.h"
```

Include dependency graph for ll_filesystem.h:



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



Functions

- `bool ll_filesystem_init ()`
- `DIRECTORY_STRUCT * ll_filesystem_dir_open (const char *path)`
- `void ll_filesystem_dir_close (DIRECTORY_STRUCT *dir)`
- `FILE_HANDLE * ll_filesystem_file_open (const char *filename)`
- `void ll_filesystem_file_close (FILE_HANDLE *handle)`
- `FILE_STATUS ll_filesystem_file_seek (FILE_HANDLE *handle, uint32_t offset)`
- `FILE_STATUS ll_filesystem_file_read (FILE_HANDLE *handle, uint8_t *buf, uint32_t size)`
- `FILE_STATUS ll_filesystem_file_write (FILE_HANDLE *handle, uint8_t *buf, uint32_t size)`

7.62 ll_filesystem.h

```

00001 /*****
00002  * Project:      discoverpixy
00003  * Website:     https://github.com/t-moe/discoverpixy
00004  * Authors:     Aaron Schmocker, Timo Lang
00005  * Institution: BFH Bern University of Applied Sciences
00006  * File:        common/lowlevel/ll_filesystem.h
00007  *
00008  * Version History:
00009  * Date      Autor Email      SHA      Changes
00010  * 2015-05-10 timolang@gmail.com e2bce8f Added filesystem module, tests and implementation for it in
00011  * 2015-05-15 timolang@gmail.com 9a16865 Added doxygen comments to filesyste, checkbox, numupdown and
00012  * screen module. And some minor changes to the other modules.
00013  *****/
00014
00015 #include "filesystem.h"
00016
00021
00027
00032
00033
00034 bool ll_filesystem_init();
00035
00036 DIRECTORY_STRUCT* ll_filesystem_dir_open(const char* path);
00037 void ll_filesystem_dir_close(DIRECTORY_STRUCT* dir);
00038 FILE_HANDLE* ll_filesystem_file_open(const char* filename);
00039 void ll_filesystem_file_close(FILE_HANDLE* handle);
00040 FILE_STATUS ll_filesystem_file_seek(
00041 FILE_HANDLE* handle, uint32_t offset);
00041 FILE_STATUS ll_filesystem_file_read(
00042 FILE_HANDLE* handle, uint8_t* buf, uint32_t size);
00042 FILE_STATUS ll_filesystem_file_write(
00043 FILE_HANDLE* handle, uint8_t* buf, uint32_t size);
00043

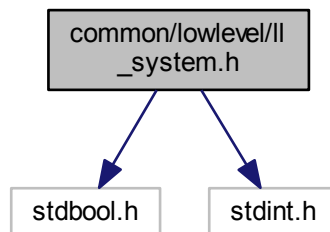
```

7.63 common/lowlevel/ll_system.h File Reference

```

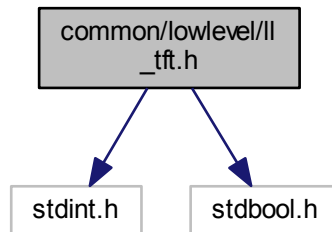
#include <stdbool.h>
#include <stdint.h>
Include dependency graph for ll_system.h:

```

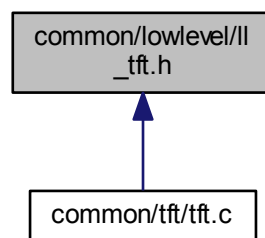


7.65 common/lowlevel/ll_tft.h File Reference

```
#include <stdint.h>
#include <stdbool.h>
Include dependency graph for ll_tft.h:
```



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



Functions

- bool [ll_tft_init](#) ()
- void [ll_tft_clear](#) (uint16_t color)
- void [ll_tft_draw_line](#) (uint16_t x1, uint16_t y1, uint16_t x2, uint16_t y2, uint16_t color)
- void [ll_tft_draw_pixel](#) (uint16_t x, uint16_t y, uint16_t color)
- void [ll_tft_draw_rectangle](#) (uint16_t x1, uint16_t y1, uint16_t x2, uint16_t y2, uint16_t color)
- void [ll_tft_fill_rectangle](#) (uint16_t x1, uint16_t y1, uint16_t x2, uint16_t y2, uint16_t color)
- void [ll_tft_draw_bitmap_unscaled](#) (uint16_t x, uint16_t y, uint16_t width, uint16_t height, const uint16_t *dat)
- void [ll_tft_draw_circle](#) (uint16_t x, uint16_t y, uint16_t r, uint16_t color)
- uint8_t [ll_tft_num_fonts](#) ()
- uint8_t [ll_tft_font_height](#) (uint8_t fontnum)
- uint8_t [ll_tft_font_width](#) (uint8_t fontnum)
- void [ll_tft_draw_char](#) (uint16_t x, uint16_t y, uint16_t color, uint16_t bgcolor, uint8_t font, char c)

7.66 ll_tft.h

```

00001 /*****
00002 * Project:      discoverpixy
00003 * Website:     https://github.com/t-moe/discoverpixy
00004 * Authors:     Aaron Schmocker, Timo Lang
00005 * Institution:  BFH Bern University of Applied Sciences
00006 * File:        common/lowlevel/ll_tft.h
00007 *
00008 * Version History:
00009 * Date      Autor Email      SHA      Changes
00010 * 2015-04-03 timolang@gmail.com 51089aa Refactored Project Structure for use with emulator
00011 * 2015-04-03 timolang@gmail.com 1f2af9f Added more tft functions to common and emulator. Fixed
00012 * 2015-04-03 timolang@gmail.com 9ald12a Refactored discovery, to use new project structure. Almost
00013 * 2015-04-03 timolang@gmail.com 1aa9194 Fixed Drawing of rects in Emulator. Got frames from pixy to
00014 * 2015-04-27 aaron@duckpond.ch aed90ef Drawcircle added (emulator)
00015 * 2015-04-27 timolang@gmail.com e249fb2 Added font support
00016 * 2015-04-27 aaron@duckpond.ch f0a6c3b Implemented init functions for gpio, fsmc and display
00017 * 2015-04-27 aaron@duckpond.ch 0b61f21 Fixed misplacement of prototypes in ll_tft.h and implemented a
00018 * 2015-05-15 timolang@gmail.com 9a16865 Added doxygen comments to filesyste, checkbox, numupdown and
00019 * screen module. And some minor changes to the other modules.
00020 *****/
00021
00022 #include <stdint.h>
00023 #include <stdbool.h>
00024
00025
00030
00036
00041
00042 // init functions
00043 bool ll_tft_init();
00044
00045 // draw functions
00046 void ll_tft_clear(uint16_t color);
00047 void ll_tft_draw_line(uint16_t x1, uint16_t y1, uint16_t x2, uint16_t y2, uint16_t color);
00048 void ll_tft_draw_pixel(uint16_t x, uint16_t y, uint16_t color);
00049 void ll_tft_draw_rectangle(uint16_t x1, uint16_t y1, uint16_t x2, uint16_t y2,
00050                             uint16_t color);
00051 void ll_tft_fill_rectangle(uint16_t x1, uint16_t y1, uint16_t x2, uint16_t y2,
00052                             uint16_t color);
00053 void ll_tft_draw_bitmap_unscaled(uint16_t x, uint16_t y, uint16_t width,
00054                                   uint16_t height, const uint16_t* dat);
00055 void ll_tft_draw_circle(uint16_t x, uint16_t y, uint16_t r, uint16_t color);
00056
00057 uint8_t ll_tft_num_fonts();
00058 uint8_t ll_tft_font_height(uint8_t fontnum);
00059 uint8_t ll_tft_font_width(uint8_t fontnum);
00060 void ll_tft_draw_char(uint16_t x, uint16_t y, uint16_t color, uint16_t bgcolor, uint8_t
00061                       font, char c);
00062
00063

```

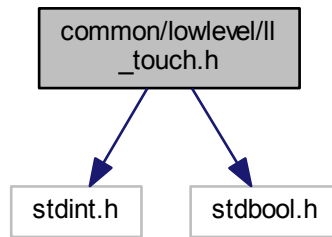
7.67 common/lowlevel/ll_touch.h File Reference

```

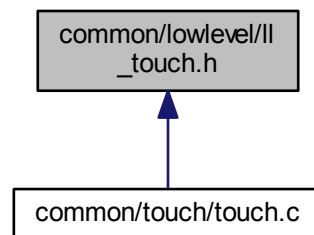
#include <stdint.h>
#include <stdbool.h>

```

Include dependency graph for ll_touch.h:



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



Functions

- bool [ll_touch_init](#) ()

7.68 ll_touch.h

```

00001 /*****
00002  * Project:      discoverpixy
00003  * Website:     https://github.com/t-moe/discoverpixy
00004  * Authors:     Aaron Schmocker, Timo Lang
00005  * Institution: BFH Bern University of Applied Sciences
00006  * File:       common/lowlevel/ll_touch.h
00007  *
00008  * Version History:
00009  * Date      Autor Email      SHA      Changes
00010  * 2015-04-27 timolang@gmail.com 259d446 Added touch support to emulator. Implemented basic touch
00011  * 2015-05-15 timolang@gmail.com 9a16865 Added doxygen comments to filesystem, checkbox, numupdown and
00012  * screen module. And some minor changes to the other modules.
00013  *****/
00014
00015 #include <stdint.h>
00016 #include <stdbool.h>
  
```

```

00017
00022
00028
00033
00034
00035 bool ll_touch_init();
00036

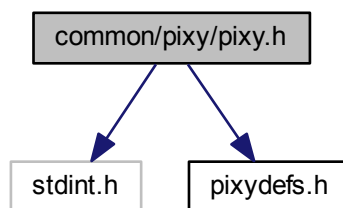
```

7.69 common/pixy/pixy.h File Reference

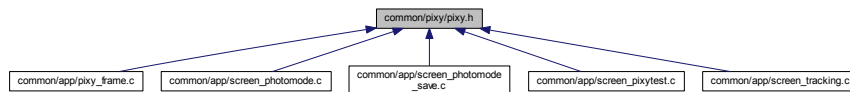
```

#include <stdint.h>
#include "pixydefs.h"
Include dependency graph for pixy.h:

```



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



Data Structures

- struct [Block](#)

Macros

- #define [PIXY_MAX_SIGNATURE](#) 7
- #define [PIXY_MIN_X](#) 0
- #define [PIXY_MAX_X](#) 319
- #define [PIXY_MIN_Y](#) 0
- #define [PIXY_MAX_Y](#) 199
- #define [PIXY_RCS_MIN_POS](#) 0
- #define [PIXY_RCS_MAX_POS](#) 1000
- #define [PIXY_RCS_CENTER_POS](#) ((PIXY_RCS_MAX_POS-PIXY_RCS_MIN_POS)/2)
- #define [PIXY_BLOCKTYPE_NORMAL](#) 0
- #define [PIXY_BLOCKTYPE_COLOR_CODE](#) 1

Functions

- [int pixy_init \(\)](#)
Creates a connection with Pixy and listens for Pixy messages.
- [int pixy_blocks_are_new \(\)](#)
Indicates when new block data from Pixy is received.
- [int pixy_get_blocks](#) (uint16_t max_blocks, struct [Block](#) *blocks)
Copies up to 'max_blocks' number of Blocks to the address pointed to by 'blocks'.
- [int pixy_service \(\)](#)
- [int pixy_command](#) (const char *name,...)
Send a command to Pixy.
- [void pixy_close \(\)](#)
Terminates connection with Pixy.
- [void pixy_error](#) (int error_code)
Send description of pixy error to stdout.
- [int pixy_led_set_RGB](#) (uint8_t red, uint8_t green, uint8_t blue)
Set color of pixy LED.
- [int pixy_led_set_max_current](#) (uint32_t current)
Set pixy LED maximum current.
- [int pixy_led_get_max_current \(\)](#)
Get pixy LED maximum current.
- [int pixy_cam_set_auto_white_balance](#) (uint8_t value)
Enable or disable pixy camera auto white balance.
- [int pixy_cam_get_auto_white_balance \(\)](#)
Get pixy camera auto white balance setting.
- [uint32_t pixy_cam_get_white_balance_value \(\)](#)
Get pixy camera white balance()
- [int pixy_cam_set_white_balance_value](#) (uint8_t red, uint8_t green, uint8_t blue)
Set pixy camera white balance.
- [int pixy_cam_set_auto_exposure_compensation](#) (uint8_t enable)
Enable or disable pixy camera auto exposure compensation.
- [int pixy_cam_get_auto_exposure_compensation \(\)](#)
Get pixy camera auto exposure compensation setting.
- [int pixy_cam_set_exposure_compensation](#) (uint8_t gain, uint16_t comp)
Set pixy camera exposure compensation.
- [int pixy_cam_get_exposure_compensation](#) (uint8_t *gain, uint16_t *comp)
Get pixy camera exposure compensation.
- [int pixy_cam_set_brightness](#) (uint8_t brightness)
Set pixy camera brightness.
- [int pixy_cam_get_brightness \(\)](#)
Get pixy camera brightness.
- [int pixy_rcs_get_position](#) (uint8_t channel)
Get pixy servo axis position.
- [int pixy_rcs_set_position](#) (uint8_t channel, uint16_t position)
Set pixy servo axis position.
- [int pixy_rcs_set_frequency](#) (uint16_t frequency)
Set pixy servo pulse width modulation (PWM) frequency.
- [int pixy_get_firmware_version](#) (uint16_t *major, uint16_t *minor, uint16_t *build)
Get pixy firmware version.

7.70 pixy.h

```

00001 //
00002 // begin license header
00003 //
00004 // This file is part of Pixy CMUcam5 or "Pixy" for short
00005 //
00006 // All Pixy source code is provided under the terms of the
00007 // GNU General Public License v2 (http://www.gnu.org/licenses/gpl-2.0.html).
00008 // Those wishing to use Pixy source code, software and/or
00009 // technologies under different licensing terms should contact us at
00010 // cmucam@cs.cmu.edu. Such licensing terms are available for
00011 // all portions of the Pixy codebase presented here.
00012 //
00013 // end license header
00014 //
00015 //
00016 #ifndef __PIXY_H__
00017 #define __PIXY_H__
00018 //
00019 #include <stdint.h>
00020 #include "pixydefs.h"
00021 //
00022 // Pixy C API //
00023 //
00024 #ifdef __cplusplus
00025 extern "C"
00026 {
00027 #endif
00028 //
00029 //
00030 //
00031 //
00032 //
00033 //
00034 //
00035 //
00036 #define PIXY_MAX_SIGNATURE          7
00037 //
00038 // Pixy x-y position values
00039 #define PIXY_MIN_X                  0
00040 #define PIXY_MAX_X                  319
00041 #define PIXY_MIN_Y                  0
00042 #define PIXY_MAX_Y                  199
00043 //
00044 // RC-servo values
00045 #define PIXY_RCS_MIN_POS            0
00046 #define PIXY_RCS_MAX_POS            1000
00047 #define PIXY_RCS_CENTER_POS         ((PIXY_RCS_MAX_POS-PIXY_RCS_MIN_POS)/2)
00048 //
00049 // Block types
00050 #define PIXY_BLOCKTYPE_NORMAL        0
00051 #define PIXY_BLOCKTYPE_COLOR_CODE    1
00052 //
00053 struct Block
00054 {
00055     /*void print(char *buf)
00056     {
00057         int i, j;
00058         char sig[6], d;
00059         bool flag;
00060         if (type==PIXY_BLOCKTYPE_COLOR_CODE)
00061         {
00062             // convert signature number to an octal string
00063             for (i=12, j=0, flag=false; i>=0; i-=3)
00064             {
00065                 d = (signature>>i)&0x07;
00066                 if (d>0 && !flag)
00067                     flag = true;
00068                 if (flag)
00069                     sig[j++] = d + '0';
00070             }
00071             sig[j] = '\0';
00072             sprintf(buf, "CC block! sig: %s (%d decimal) x: %d y: %d width: %d height: %d angle %d", sig,
signature, x, y, width, height, angle);
00073         }
00074         else // regular block. Note, angle is always zero, so no need to print
00075             sprintf(buf, "sig: %d x: %d y: %d width: %d height: %d", signature, x, y, width, height);
00076     }*/
00077 //
00078 uint16_t type;
00079 uint16_t signature;
00080 uint16_t x;
00081 uint16_t y;
00082 uint16_t width;
00083 uint16_t height;
00084 int16_t angle;
00085 };
00086 //
00087 //
00088 //
00089 //
00090 //
00091 //
00092 //
00093 //
00094 //
00095 int pixy_init();
00096 //

```

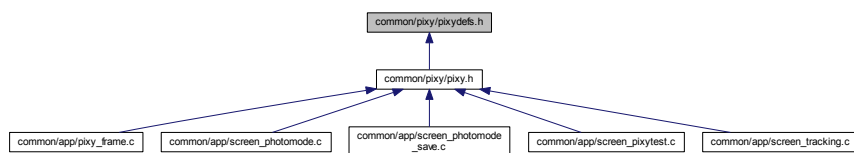
```

00104  int pixy_blocks_are_new();
00105
00121  int pixy_get_blocks(uint16_t max_blocks, struct Block * blocks);
00122
00123
00124
00125  int pixy_service();
00126
00133  int pixy_command(const char *name, ...);
00134
00138  void pixy_close();
00139
00144  void pixy_error(int error_code);
00145
00154  int pixy_led_set_RGB(uint8_t red, uint8_t green, uint8_t blue);
00155
00162  int pixy_led_set_max_current(uint32_t current);
00163
00169  int pixy_led_get_max_current();
00170
00178  int pixy_cam_set_auto_white_balance(uint8_t value);
00179
00186  int pixy_cam_get_auto_white_balance();
00187
00193  uint32_t pixy_cam_get_white_balance_value();
00194
00203  int pixy_cam_set_white_balance_value(uint8_t red, uint8_t green, uint8_t
blue);
00204
00212  int pixy_cam_set_auto_exposure_compensation(uint8_t enable);
00213
00220  int pixy_cam_get_auto_exposure_compensation();
00221
00229  int pixy_cam_set_exposure_compensation(uint8_t gain, uint16_t comp);
00230
00238  int pixy_cam_get_exposure_compensation(uint8_t * gain, uint16_t * comp)
;
00239
00246  int pixy_cam_set_brightness(uint8_t brightness);
00247
00253  int pixy_cam_get_brightness();
00254
00261  int pixy_rcs_get_position(uint8_t channel);
00262
00270  int pixy_rcs_set_position(uint8_t channel, uint16_t position);
00271
00276  int pixy_rcs_set_frequency(uint16_t frequency);
00277
00286  int pixy_get_firmware_version(uint16_t * major, uint16_t * minor, uint16_t *
build);
00287
00290 #ifdef __cplusplus
00291 }
00292 #endif
00293
00294 #endif

```

7.71 common/pixy/pixydefs.h File Reference

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



Macros

- `#define PIXY_ERROR_INVALID_PARAMETER` -150
- `#define PIXY_ERROR_CHIRP` -151

- `#define PIXY_ERROR_INVALID_COMMAND -152`
- `#define CRP_ARRAY 0x80`
- `#define CRP_FLT 0x10`
- `#define CRP_NO_COPY (0x10 | 0x20)`
- `#define CRP_NULLTERM_ARRAY (0x20 | CRP_ARRAY)`
- `#define CRP_INT8 0x01`
- `#define CRP_UINT8 0x01`
- `#define CRP_INT16 0x02`
- `#define CRP_UINT16 0x02`
- `#define CRP_INT32 0x04`
- `#define CRP_UINT32 0x04`
- `#define CRP_FLT32 (CRP_FLT | 0x04)`
- `#define CRP_FLT64 (CRP_FLT | 0x08)`
- `#define CRP_STRING (CRP_NULLTERM_ARRAY | CRP_INT8)`
- `#define CRP_TYPE_HINT 0x64`
- `#define CRP_INTS8 (CRP_INT8 | CRP_ARRAY)`
- `#define CRP_INTS16 (CRP_INT16 | CRP_ARRAY)`
- `#define CRP_INTS32 (CRP_INT32 | CRP_ARRAY)`
- `#define CRP_UINT8 CRP_INTS8`
- `#define CRP_UINT8_NO_COPY (CRP_INTS8 | CRP_NO_COPY)`
- `#define CRP_UINT16_NO_COPY (CRP_INTS16 | CRP_NO_COPY)`
- `#define CRP_UINT32_NO_COPY (CRP_INTS32 | CRP_NO_COPY)`
- `#define CRP_UINT16 CRP_INTS16`
- `#define CRP_UINT32 CRP_INTS32`
- `#define CRP_FLTS32 (CRP_FLT32 | CRP_ARRAY)`
- `#define CRP_FLTS64 (CRP_FLT64 | CRP_ARRAY)`
- `#define INT8(v) CRP_INT8, v`
- `#define UINT8(v) CRP_INT8, v`
- `#define INT16(v) CRP_INT16, v`
- `#define UINT16(v) CRP_INT16, v`
- `#define INT32(v) CRP_INT32, v`
- `#define UINT32(v) CRP_INT32, v`
- `#define FLT32(v) CRP_FLT32, v`
- `#define FLT64(v) CRP_FLT64, v`
- `#define STRING(s) CRP_STRING, s`
- `#define INTS8(len, a) CRP_INTS8, len, a`
- `#define UINTS8(len, a) CRP_INTS8, len, a`
- `#define UINTS8_NO_COPY(len) CRP_UINTS8_NO_COPY, len`
- `#define UINTS16_NO_COPY(len) CRP_UINTS16_NO_COPY, len`
- `#define UINTS32_NO_COPY(len) CRP_UINTS32_NO_COPY, len`
- `#define INTS16(len, a) CRP_INTS16, len, a`
- `#define UINTS16(len, a) CRP_INTS16, len, a`
- `#define INTS32(len, a) CRP_INTS32, len, a`
- `#define UINTS32(len, a) CRP_INTS32, len, a`
- `#define FLTS32(len, a) CRP_FLTS32, len, a`
- `#define FLTS64(len, a) CRP_FLTS64, len, a`
- `#define END 0`
- `#define END_OUT_ARGS END`
- `#define END_IN_ARGS END`

7.71.1 Macro Definition Documentation

7.71.1.1 `#define CRP_ARRAY 0x80`

Definition at line 34 of file [pixydefs.h](#).

7.71.1.2 `#define CRP_FLT 0x10`

Definition at line 35 of file [pixydefs.h](#).

7.71.1.3 `#define CRP_FLT32 (CRP_FLT | 0x04)`

Definition at line 44 of file [pixydefs.h](#).

7.71.1.4 `#define CRP_FLT64 (CRP_FLT | 0x08)`

Definition at line 45 of file [pixydefs.h](#).

7.71.1.5 `#define CRP_FLTS32 (CRP_FLT32 | CRP_ARRAY)`

Definition at line 57 of file [pixydefs.h](#).

7.71.1.6 `#define CRP_FLTS64 (CRP_FLT64 | CRP_ARRAY)`

Definition at line 58 of file [pixydefs.h](#).

7.71.1.7 `#define CRP_INT16 0x02`

Definition at line 40 of file [pixydefs.h](#).

7.71.1.8 `#define CRP_INT32 0x04`

Definition at line 42 of file [pixydefs.h](#).

7.71.1.9 `#define CRP_INT8 0x01`

Definition at line 38 of file [pixydefs.h](#).

7.71.1.10 `#define CRP_INTS16 (CRP_INT16 | CRP_ARRAY)`

Definition at line 49 of file [pixydefs.h](#).

7.71.1.11 `#define CRP_INTS32 (CRP_INT32 | CRP_ARRAY)`

Definition at line 50 of file [pixydefs.h](#).

7.71.1.12 `#define CRP_INTS8 (CRP_INT8 | CRP_ARRAY)`

Definition at line 48 of file [pixydefs.h](#).

7.71.1.13 `#define CRP_NO_COPY (0x10 | 0x20)`

Definition at line 36 of file [pixydefs.h](#).

7.71.1.14 `#define CRP_NULLTERM_ARRAY (0x20 | CRP_ARRAY)`

Definition at line 37 of file [pixydefs.h](#).

7.71.1.15 `#define CRP_STRING (CRP_NULLTERM_ARRAY | CRP_INT8)`

Definition at line 46 of file [pixydefs.h](#).

7.71.1.16 `#define CRP_TYPE_HINT 0x64`

Definition at line 47 of file [pixydefs.h](#).

7.71.1.17 `#define CRP_UINT16 0x02`

Definition at line 41 of file [pixydefs.h](#).

7.71.1.18 `#define CRP_UINT32 0x04`

Definition at line 43 of file [pixydefs.h](#).

7.71.1.19 `#define CRP_UINT8 0x01`

Definition at line 39 of file [pixydefs.h](#).

7.71.1.20 `#define CRP_UINTS16 CRP_INTS16`

Definition at line 55 of file [pixydefs.h](#).

7.71.1.21 `#define CRP_UINTS16_NO_COPY (CRP_INTS16 | CRP_NO_COPY)`

Definition at line 53 of file [pixydefs.h](#).

7.71.1.22 `#define CRP_UINTS32 CRP_INTS32`

Definition at line 56 of file [pixydefs.h](#).

7.71.1.23 `#define CRP_UINTS32_NO_COPY (CRP_INTS32 | CRP_NO_COPY)`

Definition at line 54 of file [pixydefs.h](#).

7.71.1.24 `#define CRP_UINTS8 CRP_INTS8`

Definition at line 51 of file [pixydefs.h](#).

7.71.1.25 `#define CRP_UINTS8_NO_COPY (CRP_INTS8 | CRP_NO_COPY)`

Definition at line 52 of file [pixydefs.h](#).

7.71.1.26 `#define END 0`

Definition at line 86 of file [pixydefs.h](#).

7.71.1.27 `#define END_IN_ARGS END`

Definition at line 90 of file [pixydefs.h](#).

7.71.1.28 `#define END_OUT_ARGS END`

Definition at line 89 of file [pixydefs.h](#).

7.71.1.29 `#define FLT32( v ) CRP_FLT32, v`

Definition at line 67 of file [pixydefs.h](#).

7.71.1.30 `#define FLT64( v ) CRP_FLT64, v`

Definition at line 68 of file [pixydefs.h](#).

7.71.1.31 `#define FLTS32( len, a ) CRP_FLTS32, len, a`

Definition at line 79 of file [pixydefs.h](#).

7.71.1.32 `#define FLTS64( len, a ) CRP_FLTS64, len, a`

Definition at line 80 of file [pixydefs.h](#).

7.71.1.33 `#define INT16( v ) CRP_INT16, v`

Definition at line 63 of file [pixydefs.h](#).

7.71.1.34 `#define INT32( v ) CRP_INT32, v`

Definition at line 65 of file [pixydefs.h](#).

7.71.1.35 `#define INT8( v ) CRP_INT8, v`

Definition at line 61 of file [pixydefs.h](#).

7.71.1.36 `#define INTS16( len, a ) CRP_INTS16, len, a`

Definition at line 75 of file [pixydefs.h](#).

7.71.1.37 `#define INTS32( len, a ) CRP_INTS32, len, a`

Definition at line 77 of file [pixydefs.h](#).

7.71.1.38 `#define INTS8( len, a ) CRP_INTS8, len, a`

Definition at line 70 of file [pixydefs.h](#).

7.71.1.39 `#define PIXY_ERROR_CHIRP -151`

Definition at line 31 of file [pixydefs.h](#).

7.71.1.40 `#define PIXY_ERROR_INVALID_COMMAND -152`

Definition at line 32 of file [pixydefs.h](#).

7.71.1.41 `#define PIXY_ERROR_INVALID_PARAMETER -150`

Definition at line 30 of file [pixydefs.h](#).

7.71.1.42 `#define STRING( s ) CRP_STRING, s`

Definition at line 69 of file [pixydefs.h](#).

7.71.1.43 `#define UINT16( v ) CRP_INT16, v`

Definition at line 64 of file [pixydefs.h](#).

7.71.1.44 `#define UINT32( v ) CRP_INT32, v`

Definition at line 66 of file [pixydefs.h](#).

7.71.1.45 `#define UINT8( v ) CRP_INT8, v`

Definition at line 62 of file [pixydefs.h](#).

7.71.1.46 `#define UINTS16( len, a ) CRP_INTS16, len, a`

Definition at line 76 of file [pixydefs.h](#).

7.71.1.47 `#define UINTS16_NO_COPY( len ) CRP_UINTS16_NO_COPY, len`

Definition at line 73 of file [pixydefs.h](#).

7.71.1.48 `#define UINTS32( len, a ) CRP_INTS32, len, a`

Definition at line 78 of file [pixydefs.h](#).

7.71.1.49 `#define UINTS32_NO_COPY( len ) CRP_UINTS32_NO_COPY, len`

Definition at line 74 of file [pixydefs.h](#).

7.71.1.50 `#define UINTS8( len, a ) CRP_INTS8, len, a`

Definition at line 71 of file [pixydefs.h](#).

7.71.1.51 `#define UINTS8_NO_COPY( len ) CRP_UINTS8_NO_COPY, len`

Definition at line 72 of file [pixydefs.h](#).

7.72 pixydefs.h

```

00001 //
00002 // begin license header
00003 //
00004 // This file is part of Pixy CMUcam5 or "Pixy" for short
00005 //
00006 // All Pixy source code is provided under the terms of the
00007 // GNU General Public License v2 (http://www.gnu.org/licenses/gpl-2.0.html).
00008 // Those wishing to use Pixy source code, software and/or
00009 // technologies under different licensing terms should contact us at
00010 // cmucam@cs.cmu.edu. Such licensing terms are available for
00011 // all portions of the Pixy codebase presented here.
00012 //
00013 // end license header
00014 //
00015 //
00016 #ifndef __PIXYDEFS_H__
00017 #define __PIXYDEFS_H__
00018 //
00019 // #include "libusb.h"
00020 //
00021 // #define PIXY_VID      0xB1AC
00022 // #define PIXY_DID      0xF000
00023 // #define PIXY_DFU_VID  0x1FC9
00024 // #define PIXY_DFU_DID  0x000C
00025 //
00026 // #define PIXY_ERROR_USB_IO          LIBUSB_ERROR_IO
00027 // #define PIXY_ERROR_USB_NOT_FOUND   LIBUSB_ERROR_NOT_FOUND
00028 // #define PIXY_ERROR_USB_BUSY        LIBUSB_ERROR_BUSY
00029 // #define PIXY_ERROR_USB_NO_DEVICE   LIBUSB_ERROR_NO_DEVICE
00030 #define PIXY_ERROR_INVALID_PARAMETER -150
00031 #define PIXY_ERROR_CHIRP              -151
00032 #define PIXY_ERROR_INVALID_COMMAND   -152
00033 //
00034 #define CRP_ARRAY          0x80 // bit
00035 #define CRP_FLT            0x10 // bit
00036 #define CRP_NO_COPY        (0x10 | 0x20)
00037 #define CRP_NULLTERM_ARRAY (0x20 | CRP_ARRAY) // bits
00038 #define CRP_INT8           0x01
00039 #define CRP_UINT8          0x01
00040 #define CRP_INT16          0x02
00041 #define CRP_UINT16         0x02
00042 #define CRP_INT32          0x04
00043 #define CRP_UINT32         0x04
00044 #define CRP_FLT32          (CRP_FLT | 0x04)
00045 #define CRP_FLT64          (CRP_FLT | 0x08)
00046 #define CRP_STRING         (CRP_NULLTERM_ARRAY | CRP_INT8)
00047 #define CRP_TYPE_HINT      0x64 // type hint identifier
00048 #define CRP_INTS8          (CRP_INT8 | CRP_ARRAY)
00049 #define CRP_INTS16         (CRP_INT16 | CRP_ARRAY)
00050 #define CRP_INTS32         (CRP_INT32 | CRP_ARRAY)
00051 #define CRP_INTS8          CRP_INTS8
00052 #define CRP_UINTS8_NO_COPY (CRP_INTS8 | CRP_NO_COPY)
00053 #define CRP_UINTS16_NO_COPY (CRP_INTS16 | CRP_NO_COPY)
00054 #define CRP_UINTS32_NO_COPY (CRP_INTS32 | CRP_NO_COPY)
00055 #define CRP_UINTS16        CRP_INTS16
00056 #define CRP_UINTS32        CRP_INTS32
00057 #define CRP_FLTS32         (CRP_FLT32 | CRP_ARRAY)
00058 #define CRP_FLTS64         (CRP_FLT64 | CRP_ARRAY)
00059 //
00060 // regular call args
00061 #define INT8(v)             CRP_INT8, v
00062 #define UINT8(v)            CRP_INT8, v
00063 #define INT16(v)            CRP_INT16, v
00064 #define UINT16(v)           CRP_INT16, v
00065 #define INT32(v)            CRP_INT32, v
00066 #define UINT32(v)           CRP_INT32, v
00067 #define FLT32(v)            CRP_FLT32, v
00068 #define FLT64(v)            CRP_FLT64, v
00069 #define STRING(s)           CRP_STRING, s

```

```

00070 #define INTS8(len, a)          CRP_INTS8, len, a
00071 #define UINTS8(len, a)         CRP_INTS8, len, a
00072 #define UINTS8_NO_COPY(len)    CRP_UINTS8_NO_COPY, len
00073 #define UINTS16_NO_COPY(len)   CRP_UINTS16_NO_COPY, len
00074 #define UINTS32_NO_COPY(len)   CRP_UINTS32_NO_COPY, len
00075 #define INTS16(len, a)        CRP_INTS16, len, a
00076 #define UINTS16(len, a)       CRP_INTS16, len, a
00077 #define INTS32(len, a)        CRP_INTS32, len, a
00078 #define UINTS32(len, a)       CRP_INTS32, len, a
00079 #define FLTS32(len, a)        CRP_FLTS32, len, a
00080 #define FLTS64(len, a)        CRP_FLTS64, len, a
00081
00082 #ifndef END
00083 #ifdef __x86_64__
00084 #define END (int64_t)0
00085 #else
00086 #define END 0
00087 #endif
00088 #endif
00089 #define END_OUT_ARGS END
00090 #define END_IN_ARGS END
00091
00092 #endif

```

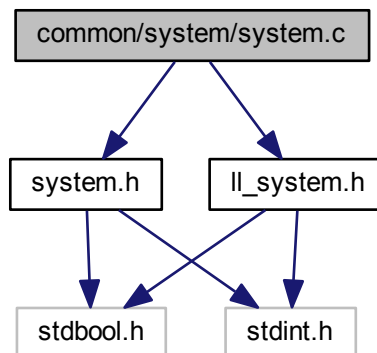
7.73 common/system/system.c File Reference

```

#include "system.h"
#include "ll_system.h"

```

Include dependency graph for system.c:



Functions

- bool [system_init](#) ()
- void [system_delay](#) (uint32_t msec)
- void [system_process](#) ()
- void [system_toggle_led](#) ()

7.74 system.c

```

00001 /*****
00002  * Project:      discoverpixy
00003  * Website:     https://github.com/t-moe/discoverpixy

```

```

00004 * Authors:      Aaron Schmocker, Timo Lang
00005 * Institution:   BFH Bern University of Applied Sciences
00006 * File:         common/system/system.c
00007 *
00008 * Version History:
00009 * Date           Autor Email           SHA           Changes
00010 * 2015-04-03     timolang@gmail.com    1f2af9f Added more tft functions to common and emulator. Fixed
00011 *               timolang@gmail.com    cab8609 Integrated pixy into emulator. Pixy is no longer in the common/
00012 *               timolang@gmail.com    3d1e4b2 Simplified code a bit. Emulator does not work stable when
00013 *               timolang@gmail.com    0858b0d Fixed some bugs when receiving large data.
00014 *
00015 *****
00016 *****/
00017 #include "system.h"
00018 #include "ll_system.h"
00019
00020
00021 bool system_init()
00022 {
00023     return ll_system_init();
00024 }
00025
00026 void system_delay(uint32_t msec)
00027 {
00028     ll_system_delay(msec);
00029 }
00030
00031 void system_process()
00032 {
00033     ll_system_process();
00034 }
00035
00036 void system_toggle_led()
00037 {
00038     ll_system_toggle_led();
00039 }

```

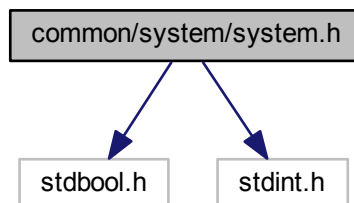
7.75 common/system/system.h File Reference

```

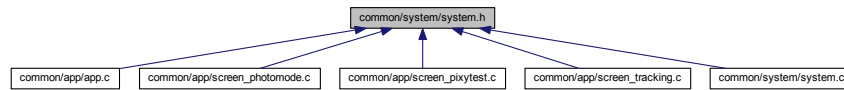
#include <stdbool.h>
#include <stdint.h>

```

Include dependency graph for system.h:



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



Functions

- bool `system_init()`
- void `system_delay(uint32_t msec)`
- void `system_process()`
- void `system_toggle_led()`

7.76 system.h

```

00001 /*****
00002  * Project:      discoverpixy
00003  * Website:     https://github.com/t-moe/discoverpixy
00004  * Authors:     Aaron Schmocker, Timo Lang
00005  * Institution: BFH Bern University of Applied Sciences
00006  * File:        common/system/system.h
00007  *
00008  * Version History:
00009  * Date          Autor Email          SHA          Changes
00010  * 2015-04-03    timolang@gmail.com    1f2af9f Added more tft functions to common and emulator. Fixed
00011  *              timolang@gmail.com    cab8609 Integrated pixy into emulator. Pixy is no longer in the common/
00012  *              timolang@gmail.com    3dle4b2 Simplified code a bit. Emulator does not work stable when
00013  *              timolang@gmail.com    0858b0d Fixed some bugs when receiving large data.
00014  *              timolang@gmail.com    08d9fe0 More work on doxygen module structure
00015  *              timolang@gmail.com    9a16865 Added doxygen comments to filesystem, checkbox, numupdown and
00016  *              screen module. And some minor changes to the other modules.
00017  *
00018  *****/
00019 #ifndef SYSTEM_H
00020 #define SYSTEM_H
00021
00022 #include <stdbool.h>
00023 #include <stdint.h>
00024
00025
00026
00027
00028
00029
00030
00031
00032
00033
00034
00035
00036 bool system_init();
00037
00038
00039
00040
00041
00042 void system_delay(uint32_t msec);
00043
00044
00045
00046
00047 void system_process();
00048
00049
00050
00051
00052 void system_toggle_led();
00053
00054
00055
00056 #endif /* SYSTEM_H */

```

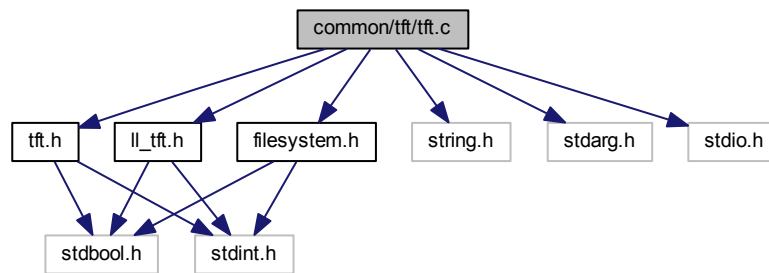
7.77 common/tft/tft.c File Reference

```

#include "tft.h"
#include "ll_tft.h"
#include <string.h>
#include <stdarg.h>
#include <stdio.h>
#include "filesystem.h"

```

Include dependency graph for tft.c:



Functions

- bool `tft_init` ()
- void `tft_clear` (uint16_t color)
- void `tft_draw_line` (uint16_t x1, uint16_t y1, uint16_t x2, uint16_t y2, uint16_t color)
- void `tft_draw_pixel` (uint16_t x, uint16_t y, uint16_t color)
- void `tft_draw_rectangle` (uint16_t x1, uint16_t y1, uint16_t x2, uint16_t y2, uint16_t color)
- void `tft_fill_rectangle` (uint16_t x1, uint16_t y1, uint16_t x2, uint16_t y2, uint16_t color)
- void `tft_draw_bitmap_unscaled` (uint16_t x, uint16_t y, uint16_t width, uint16_t height, const uint16_t *dat)
- void `tft_draw_circle` (uint16_t x, uint16_t y, uint16_t r, uint16_t color)
- uint8_t `tft_num_fonts` ()
- uint8_t `tft_font_height` (uint8_t fontnum)
- uint8_t `tft_font_width` (uint8_t fontnum)
- void `tft_print_line` (uint16_t x, uint16_t y, uint16_t color, uint16_t bgcolor, uint8_t font, const char *text)
- void `tft_print_formatted` (uint16_t x, uint16_t y, uint16_t color, uint16_t bgcolor, uint8_t font, const char *format,...)
- bool `tft_draw_bitmap_file_unscaled` (uint16_t x, uint16_t y, const char *filename)

7.78 tft.c

```

00001 /*****
00002  * Project:      discoverpixy
00003  * Website:     https://github.com/t-moe/discoverpixy
00004  * Authors:     Aaron Schmocker, Timo Lang
00005  * Institution: BFH Bern University of Applied Sciences
00006  * File:        common/tft/tft.c
00007  *
00008  * Version History:
00009  * Date      Autor Email      SHA      Changes
00010  * 2015-04-03 timolang@gmail.com 51089aa Refactored Project Structure for use with emulator
00011  * 2015-04-03 timolang@gmail.com 1f2af9f Added more tft functions to common and emulator. Fixed
00012  * 2015-04-03 timolang@gmail.com 1aa9194 Fixed Drawing of rects in Emulator. Got frames from pixy to
00013  * 2015-04-27 aaron@duckpond.ch aed90ef Drawcircle added (emulator)
00014  * 2015-04-27 timolang@gmail.com e249fb2 Added font support
00015  * 2015-04-30 timolang@gmail.com 76ea9d8 Added num up down support.
00016  * 2015-05-10 timolang@gmail.com b6ab7c8 Fixed compiler warning in tft and screen module.
00017  * 2015-05-15 timolang@gmail.com b08a897 Added tft method to draw a bmp from filesystem. Added another
00018  * 2015-05-17 timolang@gmail.com 2d46336 Improved comments in implementation of button, checkbox,
00019  * numupdown, tft, touch and screen modules/submodules.
00020  *****/

```

```

00021
00022 #include "tft.h"
00023 #include "ll_tft.h"
00024 #include <string.h>
00025 #include <stdarg.h>
00026 #include <stdio.h>
00027 #include "filesystem.h"
00028
00029 /* The idea is as follows:
00030  * Most of the tft_* functions can be forwarded to the lowlevel implementation.
00031  * The exceptions are commented below.
00032  * Make sure to have a look at the doxygen comments for the lowlevel functions and for the tft_* functions
00033  */
00034
00035 /* Possible improvements:
00036  * For formatted printing implement putchar, instead of writing into a buffer and drawing that buffer
    afterwards
00037  */
00038
00039 bool tft_init()
00040 {
00041     return ll_tft_init();
00042 }
00043
00044
00045 void tft_clear(uint16_t color)
00046 {
00047     ll_tft_clear(color);
00048 }
00049
00050 void tft_draw_line(uint16_t x1, uint16_t y1, uint16_t x2, uint16_t y2, uint16_t color)
00051 {
00052     ll_tft_draw_line(x1, y1, x2, y2, color);
00053 }
00054
00055
00056 void tft_draw_pixel(uint16_t x, uint16_t y, uint16_t color)
00057 {
00058     ll_tft_draw_pixel(x, y, color);
00059 }
00060
00061 void tft_draw_rectangle(uint16_t x1, uint16_t y1, uint16_t x2, uint16_t y2, uint16_t
    color)
00062 {
00063     //could be implemented with 4 lines instead of introducing a ll func
00064     ll_tft_draw_rectangle(x1, y1, x2, y2, color);
00065 }
00066
00067 void tft_fill_rectangle(uint16_t x1, uint16_t y1, uint16_t x2, uint16_t y2, uint16_t
    color)
00068 {
00069     ll_tft_fill_rectangle(x1, y1, x2, y2, color);
00070 }
00071
00072 void tft_draw_bitmap_unscaled(uint16_t x, uint16_t y, uint16_t width, uint16_t
    height, const uint16_t* dat)
00073 {
00074     ll_tft_draw_bitmap_unscaled(x, y, width, height, dat);
00075 }
00076
00077 void tft_draw_circle(uint16_t x, uint16_t y, uint16_t r, uint16_t color)
00078 {
00079     ll_tft_draw_circle(x, y, r, color);
00080 }
00081
00082 uint8_t tft_num_fonts()
00083 {
00084     return ll_tft_num_fonts();
00085 }
00086
00087 uint8_t tft_font_height(uint8_t fontnum)
00088 {
00089     return ll_tft_font_height(fontnum);
00090 }
00091
00092 uint8_t tft_font_width(uint8_t fontnum)
00093 {
00094     return ll_tft_font_width(fontnum);
00095 }
00096
00097 //Print line can be done with multiple calls to draw_char
00098 void tft_print_line(uint16_t x, uint16_t y, uint16_t color, uint16_t bgcolor, uint8_t font,
    const char* text)
00099 {
00100     if (font >= ll_tft_num_fonts()) {
00101         return; //invalid font index
00102     }

```

```

00103
00104     for (int i = 0; i < strlen(text); i++) { //for each char in the line
00105         ll_tft_draw_char(x, y, color, bgcolor, font, text[i]); //draw the char
00106         x += ll_tft_font_width(font); //and increase the x position
00107     }
00108 }
00109
00110 //Printing a formatted line can be done by printing the line in a buffer using "sprintf" and then calling
00111 //print_line
00112 void tft_print_formatted(uint16_t x, uint16_t y, uint16_t color, uint16_t bgcolor,
00113     uint8_t font, const char* format, ...)
00114 {
00115     static char buffer[128]; //buffer to save the formatted text into
00116     //Since we have variable arguments, we need to forward them. We have to use vsprintf instead of sprintf
00117     //for that.
00118     va_list args;
00119     va_start(args, format); //start the varg-list
00120     vsprintf(buffer, format, args); //let vsprintf render the formatted string
00121     tft_print_line(x, y, color, bgcolor, font, buffer); //print the string as normal text
00122     va_end(args); //end the varg-list
00123 }
00124
00125 bool tft_draw_bitmap_file_unscaled(uint16_t x, uint16_t y, const char*
00126     filename)
00127 {
00128     //This method reads a .bmp file from the filesystem and tries to draw it.
00129     //Note: The bmp implementation is not complete, it has some limitations and it makes assumptions. See
00130     //doxygen comment for this method.
00131     //Source Copied and adapted from: http://stackoverflow.com/a/17040962/2606757
00132
00133     FILE_HANDLE* file = filesystem_file_open(filename); //try to open the
00134     file
00135
00136     if (file == NULL) { //file opening failed
00137         return false;
00138     }
00139
00140     unsigned char info[54];
00141
00142     if (filesystem_file_read(file, info, 54) != F_OK) { //try to read the 54 byte
00143         header
00144         filesystem_file_close(file);
00145         return false; //reading the header failed
00146     }
00147
00148     // extract image height and width from header
00149     uint32_t width = *(uint32_t*)&info[18]; //width in pixel
00150     uint32_t height = *(uint32_t*)&info[22]; //height in pixel
00151     uint16_t depth = *(uint16_t*)&info[28]; //bit's per pixel (color depth)
00152     depth /= 8; //we want the number of bytes per pixel
00153
00154     filesystem_file_seek(file, *(uint32_t*)&info[10]); //seek to the place where img
00155     data begins
00156
00157     uint32_t row_padded = (width * depth + 3) & (~3); //row size must be aligned to 4 bytes
00158
00159     unsigned char data [row_padded]; //allocate space for one row (incl. padding)
00160
00161     for (int i = 0; i < height; i++) { //for each row
00162         filesystem_file_read(file, data, row_padded); //read row into buffer
00163
00164         for (int j = 0; j < width * depth; j += depth) { //for each pixel
00165             unsigned char a, r, g, b;
00166
00167             if (depth == 4) { //a,r,g,b 8bit each
00168                 a = data[j];
00169                 r = data[j + 1];
00170                 g = data[j + 2];
00171                 b = data[j + 3];
00172             } else if (depth == 3) { // b,g,r, 8bit each
00173                 a = 255;
00174                 r = data[j + 2];
00175                 g = data[j + 1];
00176                 b = data[j];
00177             }
00178
00179             if (a != 0) {
00180                 //bmp's are stored "bottom-up", so we start drawing at the bottom
00181                 tft_draw_pixel(x + j / depth, y + height - 1 - i,
00182                     RGB(r, g, b));
00183             }
00184         }
00185     }
00186
00187     filesystem_file_close(file);
00188 }

```


- `uint8_t tft_font_height` (`uint8_t fontnum`)
- `uint8_t tft_font_width` (`uint8_t fontnum`)
- `void tft_print_line` (`uint16_t x`, `uint16_t y`, `uint16_t color`, `uint16_t bgcolor`, `uint8_t font`, `const char *text`)
- `void tft_print_formatted` (`uint16_t x`, `uint16_t y`, `uint16_t color`, `uint16_t bgcolor`, `uint8_t font`, `const char *format`,...)

7.80 tft.h

```

00001 /*****
00002 * Project:      discoverpixy
00003 * Website:      https://github.com/t-moe/discoverpixy
00004 * Authors:      Aaron Schmocker, Timo Lang
00005 * Institution:  BFH Bern University of Applied Sciences
00006 * File:         common/tft/tft.h
00007 *
00008 * Version History:
00009 * Date          Autor Email          SHA          Changes
00010 * 2015-04-03    timolang@gmail.com    51089aa    Refactored Project Structure for use with emulator
00011 * 2015-04-03    timolang@gmail.com    1f2af9f    Added more tft functions to common and emulator. Fixed
00012 * 2015-04-03    timolang@gmail.com    1aa9194    Fixed Drawing of rects in Emulator. Got frames from pixy to
00013 * 2015-04-27    aaron@duckpond.ch     aed90ef    Drawcircle added (emulator)
00014 * 2015-04-27    timolang@gmail.com    e249fb2    Added font support
00015 * 2015-04-30    timolang@gmail.com    76ea9d8    Added num up down support.
00016 * 2015-05-04    aaron@duckpond.ch     c224d40    Changed display init
00017 * 2015-05-10    timolang@gmail.com    21edc56    Added doxyfile (doxygen) for the common folder. Started with
00018 * 2015-05-11    timolang@gmail.com    a175a2f    Added doxygen docu for touch module
00019 * 2015-05-11    timolang@gmail.com    08d9fe0    More work on doxygen module structure
00020 * 2015-05-12    timolang@gmail.com    1402598    Added doxygen stuff for button module and some minor changes to
00021 * 2015-05-15    timolang@gmail.com    9a16865    Added doxygen comments to filesystem, checkbox, numupdown and
00022 * 2015-05-15    timolang@gmail.com    b08a897    Added tft method to draw a bmp from filesystem. Added another
00023 * font to emulator.
00024 *****/
00025
00026 #ifndef TFT_H
00027 #define TFT_H
00028
00029 #include<stdbool.h>
00030 #include<stdint.h>
00031
00032
00033
00034
00035 #define RGB(r,g,b) (((r) & 0xF8) << 8) | (((g) & 0xFC) << 3) | (((b) & 0xF8) >> 3))
00036
00037 #define RED RGB(255,0,0)
00038 #define GREEN RGB(0,255,0)
00039 #define BLUE RGB(0,0,255)
00040 #define WHITE 0xF7BE
00041 #define BLACK RGB(0,0,0)
00042
00043 #define HEX(h) (RGB(((h)>>16),((h)>>8),(h)))
00044
00045 #define TRANSPARENT ((uint16_t)0x80C2)
00046
00047 bool tft_init();
00048
00049 void tft_clear(uint16_t color);
00050
00051 void tft_draw_line(uint16_t x1, uint16_t y1, uint16_t x2, uint16_t y2, uint16_t color);
00052
00053 void tft_draw_pixel(uint16_t x, uint16_t y, uint16_t color);
00054
00055 void tft_draw_rectangle(uint16_t x1, uint16_t y1, uint16_t x2, uint16_t y2, uint16_t
00056 color);
00057
00058 void tft_fill_rectangle(uint16_t x1, uint16_t y1, uint16_t x2, uint16_t y2, uint16_t
00059 color);
00060
00061 void tft_draw_bitmap_unscaled(uint16_t x, uint16_t y, uint16_t width, uint16_t
00062 height, const uint16_t* dat);
00063
00064 bool tft_draw_bitmap_file_unscaled(uint16_t x, uint16_t y, const char*
00065 filename);

```

```

00141
00142
00150 void tft_draw_circle(uint16_t x, uint16_t y, uint16_t r, uint16_t color);
00151
00156 uint8_t tft_num_fonts();
00157
00163 uint8_t tft_font_height(uint8_t fontnum);
00164
00170 uint8_t tft_font_width(uint8_t fontnum);
00171
00181 void tft_print_line(uint16_t x, uint16_t y, uint16_t color, uint16_t bgcolor, uint8_t font,
    const char* text);
00182
00193 void tft_print_formatted(uint16_t x, uint16_t y, uint16_t color, uint16_t bgcolor,
    uint8_t font, const char* format, ...);
00194
00197 #endif /* TFT_H */

```

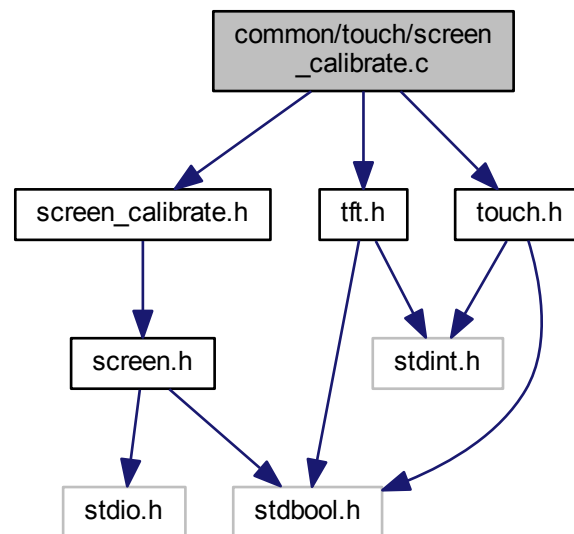
7.81 common/touch/screen_calibrate.c File Reference

```
#include "screen_calibrate.h"
```

```
#include "tft.h"
```

```
#include "touch.h"
```

Include dependency graph for screen_calibrate.c:



Functions

- static void `enter` (void *`screen`)
- static void `leave` (void *`screen`)
- static void `update` (void *`screen`)
- `SCREEN_STRUCT` * `get_screen_calibrate` ()

Variables

- volatile bool `calibration`

- static [SCREEN_STRUCT](#) `screen`

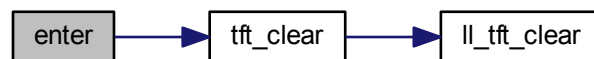
7.81.1 Function Documentation

7.81.1.1 static void enter (void * *screen*) [static]

Definition at line 23 of file [screen_calibrate.c](#).

```
00024 {
00025     tft_clear(BLACK);
00026 }
```

Here is the call graph for this function:



7.81.1.2 static void leave (void * *screen*) [static]

Definition at line 28 of file [screen_calibrate.c](#).

```
00029 {
00030
00031 }
```

7.81.1.3 static void update (void * *screen*) [static]

Definition at line 33 of file [screen_calibrate.c](#).

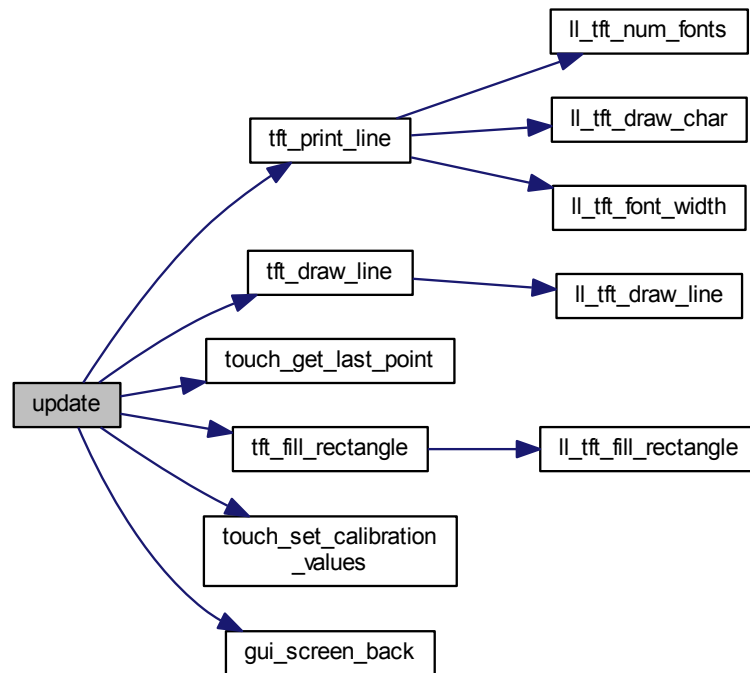
```
00034 {
00035     int x1, y1, x2, y2, dx, dy;
00036
00037     tft_print_line(50, 50, WHITE, BLACK, 1, "Calibration:");
00038     tft_print_line(50, 120, WHITE, BLACK, 0, "Hit the markers exactly!");
00039     //-----First Point-----
00040     tft_draw_line(CCENTER, CBEGIN, CCENTER,
00041     CEND, WHITE); //Draw Cross
00042     tft_draw_line(CBEGIN, CCENTER, CEND, CCENTER,
00043     WHITE); //Draw Cross
00044     calibration = 1; //TouchX + TouchY Values will not be converted to Pixels
00045     while (calibration); //Wait on PenUp
00046
00047     POINT_STRUCT p1 = touch_get_last_point();
00048     x1 = p1.x;
00049     y1 = p1.y;
00050     tft_fill_rectangle(CBEGIN, CBEGIN, CEND,
00051     CEND, BLACK); //Clear Cross
00052     //-----Second Point-----
00053     tft_draw_line(DWIDTH - CCENTER, DHEIGHT -
00054     CBEGIN, DWIDTH - CCENTER, DHEIGHT - CEND, WHITE);
00055     tft_draw_line(DWIDTH - CBEGIN, DHEIGHT -
00056     CCENTER, DWIDTH - CEND, DHEIGHT - CCENTER, WHITE);
00057     calibration = 1;
00058 }
```

```

00057     while (calibration);
00058
00059     POINT_STRUCT p2 = touch_get_last_point();
00060     x2 = p2.x;
00061     y2 = p2.y;
00062     tft_fill_rectangle(DWIDTH - CBEGIN, DHEIGHT -
CBEGIN, DWIDTH - CEND, DHEIGHT - CEND, BLACK);
00063
00064     //-----Third Point-----
00065     tft_draw_line(CCENTER, DHEIGHT - CBEGIN,
CCENTER, DHEIGHT - CEND, WHITE);
00066     tft_draw_line(CBEGIN, DHEIGHT - CCENTER,
CEND, DHEIGHT - CCENTER, WHITE);
00067     calibration = 1;
00068
00069     while (calibration);
00070
00071     POINT_STRUCT p3 = touch_get_last_point();
00072     x1 += p3.x; //Add(!) values. We'll build the average later
00073     y2 += p3.y;
00074     tft_fill_rectangle(CBEGIN, DHEIGHT - CBEGIN,
CEND, DHEIGHT - CEND, BLACK);
00075
00076     //-----4. Point-----
00077     tft_draw_line(DWIDTH - CCENTER, CBEGIN,
DWIDTH - CCENTER, CEND, WHITE);
00078     tft_draw_line(DWIDTH - CBEGIN, CCENTER,
DWIDTH - CEND, CCENTER, WHITE);
00079     calibration = 1;
00080
00081     while (calibration);
00082
00083     POINT_STRUCT p4 = touch_get_last_point();
00084     x2 += p4.x;
00085     y1 += p4.y;
00086     tft_fill_rectangle(DWIDTH - CBEGIN, CBEGIN,
DWIDTH - CEND, CEND, BLACK);
00087     //-----Calculation-----
00088     x1++; //Add 1 and divide by 2 later = +0.5 (for correct rounding)
00089     y1++;
00090     x2++;
00091     y2++;
00092     x1 >>= 1; //Divide by 2
00093     y1 >>= 1;
00094     x2 >>= 1;
00095     y2 >>= 1;
00096     dx = (x2 - x1); //Build the Difference
00097     dy = (y2 - y1);
00098
00099     touch_set_calibration_values(x1, dx, y1, dy);
00100     tft_print_line(50, 120, WHITE, BLACK, 0, "Calibration Done. Press anywhere");
00101
00102     calibration = 1;
00103
00104     while (calibration);
00105
00106     gui_screen_back();
00107
00108 }

```

Here is the call graph for this function:



7.81.2 Variable Documentation

7.81.2.1 volatile bool calibration

Definition at line 40 of file [touch.c](#).

7.81.2.2 SCREEN_STRUCT screen [static]

Initial value:

```

= {
    enter,
    leave,
    update
}

```

Definition at line 111 of file [screen_calibrate.c](#).

7.82 screen_calibrate.c

```

00001 /*****
00002  * Project:      discoverpixy
00003  * Website:     https://github.com/t-moe/discoverpixy
00004  * Authors:     Aaron Schmocker, Timo Lang
00005  * Institution: BFH Bern University of Applied Sciences
00006  * File:        common/touch/screen_calibrate.c
00007  *

```

```

00008 * Version History:
00009 * Date      Autor Email      SHA      Changes
00010 * 2015-06-01 timolang@gmail.com 06227da Added calibrate screen (WIP). fixed bug in emulator drawing.
00011 * 2015-06-01 timolang@gmail.com eb573bc Finalized calibration. Fixed a bug in screen module.
00012 *
00013 *****
00014 *****/
00015 #include "screen_calibrate.h"
00016 #include "tft.h"
00017 #include "touch.h"
00018
00019
00020 extern volatile bool calibration; //from touch.c
00021
00022
00023 static void enter(void* screen)
00024 {
00025     tft_clear(BLACK);
00026 }
00027
00028 static void leave(void* screen)
00029 {
00030
00031 }
00032
00033 static void update(void* screen)
00034 {
00035     int x1, y1, x2, y2, dx, dy;
00036
00037
00038     tft_print_line(50, 50, WHITE, BLACK, 1, "Calibration:");
00039     tft_print_line(50, 120, WHITE, BLACK, 0, "Hit the markers exactly!");
00040     //-----First Point-----
00041     tft_draw_line(CCENTER, CBEGIN, CCENTER,
00042 CEND, WHITE); //Draw Cross
00043     tft_draw_line(CBEGIN, CCENTER, CEND, CCENTER,
00044 WHITE); //Draw Cross
00045     calibration = 1; //TouchX + TouchY Values will not be converted to Pixels
00046
00047     while (calibration); //Wait on PenUp
00048
00049     POINT_STRUCT p1 = touch_get_last_point();
00050     x1 = p1.x;
00051     y1 = p1.y;
00052     tft_fill_rectangle(CBEGIN, CBEGIN, CEND,
00053 CEND, BLACK); //Clear Cross
00054
00055     //-----Second Point-----
00056     tft_draw_line(DWIDTH - CCENTER, DHEIGHT -
00057 CBEGIN, DWIDTH - CCENTER, DHEIGHT - CEND, WHITE);
00058     tft_draw_line(DWIDTH - CBEGIN, DHEIGHT -
00059 CCENTER, DWIDTH - CEND, DHEIGHT - CCENTER, WHITE);
00060     calibration = 1;
00061
00062     while (calibration);
00063
00064     POINT_STRUCT p2 = touch_get_last_point();
00065     x2 = p2.x;
00066     y2 = p2.y;
00067     tft_fill_rectangle(DWIDTH - CBEGIN, DHEIGHT -
00068 CBEGIN, DWIDTH - CEND, DHEIGHT - CEND, BLACK);
00069
00070     //-----Third Point-----
00071     tft_draw_line(CCENTER, DHEIGHT - CBEGIN,
00072 CCENTER, DHEIGHT - CEND, WHITE);
00073     tft_draw_line(CBEGIN, DHEIGHT - CCENTER,
00074 CEND, DHEIGHT - CCENTER, WHITE);
00075     calibration = 1;
00076
00077     while (calibration);
00078
00079     POINT_STRUCT p3 = touch_get_last_point();
00080     x1 += p3.x; //Add(!) values. We'll build the average later
00081     y2 += p3.y;
00082     tft_fill_rectangle(CBEGIN, DHEIGHT - CBEGIN,
00083 CEND, DHEIGHT - CEND, BLACK);
00084
00085     //-----4. Point-----
00086     tft_draw_line(DWIDTH - CCENTER, CBEGIN,
00087 DWIDTH - CCENTER, CEND, WHITE);
00088     tft_draw_line(DWIDTH - CBEGIN, CCENTER,
00089 DWIDTH - CEND, CCENTER, WHITE);
00090     calibration = 1;
00091
00092     while (calibration);
00093

```

```

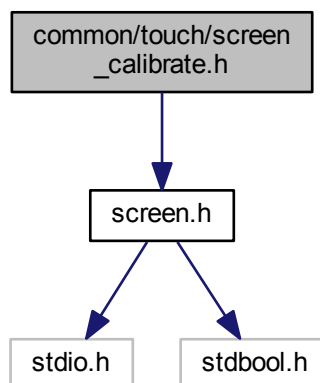
00083     POINT_STRUCT p4 = touch_get_last_point();
00084     x2 += p4.x;
00085     y1 += p4.y;
00086     tft_fill_rectangle(DWIDTH - CBEGIN, CBEGIN,
DWIDTH - CEND, CEND, BLACK);
00087     //-----Calculation-----
00088     x1++; //Add 1 and divide by 2 later = +0.5 (for correct rounding)
00089     y1++;
00090     x2++;
00091     y2++;
00092     x1 >>= 1; //Divide by 2
00093     y1 >>= 1;
00094     x2 >>= 1;
00095     y2 >>= 1;
00096     dx = (x2 - x1); //Build the Difference
00097     dy = (y2 - y1);
00098
00099     touch_set_calibration_values(x1, dx, y1, dy);
00100     tft_print_line(50, 120, WHITE, BLACK, 0, "Calibration Done. Press anywhere");
00101
00102     calibration = 1;
00103
00104     while (calibration);
00105
00106     gui_screen_back();
00107 }
00108 }
00109
00110
00111 static SCREEN_STRUCT screen = {
00112     enter,
00113     leave,
00114     update
00115 };
00116
00117
00118 SCREEN_STRUCT* get_screen_calibrate()
00119 {
00120     return &screen;
00121 }

```

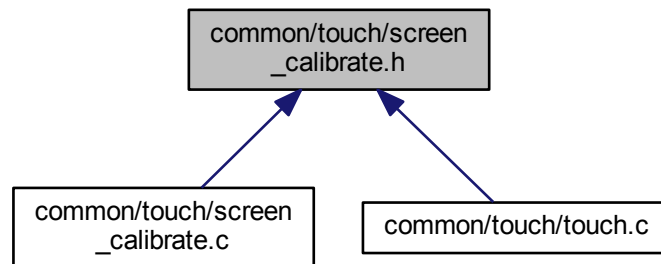
7.83 common/touch/screen_calibrate.h File Reference

#include "screen.h"

Include dependency graph for screen_calibrate.h:



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



Macros

- `#define CCENTER 20`
- `#define CLENGTH 10`
- `#define CBEGIN (CCENTER-CLENGTH/2)`
- `#define CEND (CCENTER + CLENGTH/2)`
- `#define DWIDTH 320`
- `#define DHEIGHT 240`

Functions

- `SCREEN_STRUCT * get_screen_calibrate ()`

7.83.1 Macro Definition Documentation

7.83.1.1 `#define CBEGIN (CCENTER-CLENGTH/2)`

Definition at line 44 of file [screen_calibrate.h](#).

7.83.1.2 `#define CCENTER 20`

Definition at line 42 of file [screen_calibrate.h](#).

7.83.1.3 `#define CEND (CCENTER + CLENGTH/2)`

Definition at line 45 of file [screen_calibrate.h](#).

7.83.1.4 `#define CLENGTH 10`

Definition at line 43 of file [screen_calibrate.h](#).

7.83.1.5 `#define DHEIGHT 240`

Definition at line 47 of file [screen_calibrate.h](#).

7.83.1.6 #define DWIDTH 320

Definition at line 46 of file [screen_calibrate.h](#).

7.84 screen_calibrate.h

```

00001 /*****
00002 * Project:      discoverpixy
00003 * Website:     https://github.com/t-moe/discoverpixy
00004 * Authors:     Aaron Schmocker, Timo Lang
00005 * Institution:  BFH Bern University of Applied Sciences
00006 * File:        common/touch/screen_calibrate.h
00007 *
00008 * Version History:
00009 * Date      Autor Email      SHA      Changes
00010 * 2015-06-01 timolang@gmail.com 06227da Added calibrate screen (WIP). fixed bug in emulator drawing.
00011 * 2015-06-01 timolang@gmail.com eb573bc Finalized calibration. Fixed a bug in screen module.
00012 *
00013 *****/
00014
00015 #include "screen.h"
00016
00021
00022
00028
00029
00035 SCREEN_STRUCT* get_screen_calibrate();
00036
00039
00040
00041 //TODO: Move this define to a common accessible, but private header file (they are used by
        screen_calibrate.c and touch.c)
00042 #define CCENTER 20 //Pixel Distance from Sides for Calibration Cross
00043 #define CLENGTH 10 //Length of the Calibration Cross Lines
00044 #define CBEGIN (CCENTER-CLENGTH/2)
00045 #define CEND (CCENTER + CLENGTH/2)
00046 #define DWIDTH 320 //TODO: move define to tft module or make a function out of it
00047 #define DHEIGHT 240 //TODO: move define to tft module or make a function out of it

```

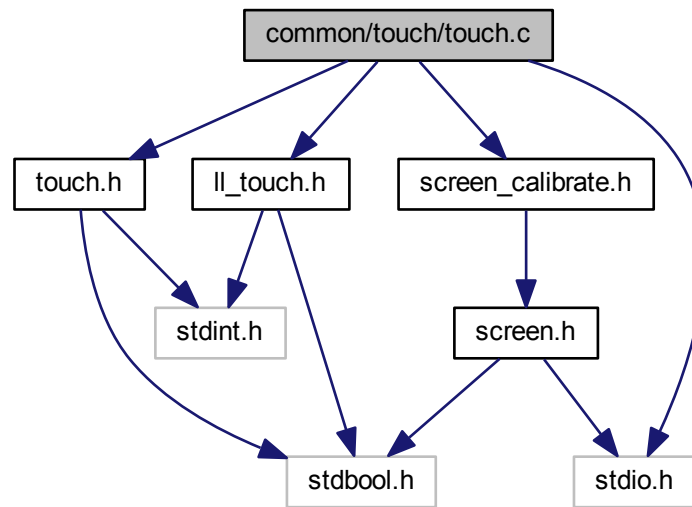
7.85 common/touch/touch.c File Reference

```

#include "touch.h"
#include "ll_touch.h"
#include "screen_calibrate.h"
#include <stdio.h>

```

Include dependency graph for touch.c:



Macros

- `#define NUM_AREAS 50`

Functions

- void `touch_set_calibration_values` (int xs, int dx, int ys, int dy)
- bool `touch_init` ()
- void `touch_set_value_convert_mode` (bool uc)
- bool `touch_add_raw_event` (uint16_t touchX, uint16_t touchY, TOUCH_STATE state)
- bool `touch_have_empty` (unsigned char num)
- bool `touch_register_area` (TOUCH_AREA_STRUCT *area)
- void `touch_unregister_area` (TOUCH_AREA_STRUCT *area)
- POINT_STRUCT `touch_get_last_point` ()

Variables

- TOUCH_AREA_STRUCT * `areas` [NUM_AREAS] = {NULL}
- volatile POINT_STRUCT `pos`
- volatile TOUCH_STATE `oldState` = TOUCH_UP
- volatile bool `calibration` = false
- bool `use_calibration` = false
- int `cal_xs` = 10
- int `cal_dx` = 100
- int `cal_ys` = 10
- int `cal_dy` = 100

7.85.1 Macro Definition Documentation

7.85.1.1 #define NUM_AREAS 50

Definition at line 35 of file [touch.c](#).

7.85.2 Variable Documentation

7.85.2.1 TOUCH_AREA_STRUCT* areas[NUM_AREAS] = {NULL}

Definition at line 36 of file [touch.c](#).

7.85.2.2 int cal_dx = 100

Definition at line 46 of file [touch.c](#).

7.85.2.3 int cal_dy = 100

Definition at line 48 of file [touch.c](#).

7.85.2.4 int cal_xs = 10

Definition at line 45 of file [touch.c](#).

7.85.2.5 int cal_ys = 10

Definition at line 47 of file [touch.c](#).

7.85.2.6 volatile bool calibration = false

Definition at line 40 of file [touch.c](#).

7.85.2.7 volatile TOUCH_STATE oldState = TOUCH_UP

Definition at line 39 of file [touch.c](#).

7.85.2.8 volatile POINT_STRUCT pos

Definition at line 38 of file [touch.c](#).

7.85.2.9 bool use_calibration = false

Definition at line 42 of file [touch.c](#).

7.86 touch.c

```
00001 /*****
00002  * Project:      discoverpixy
00003  * Website:     https://github.com/t-moe/discoverpixy
00004  * Authors:     Aaron Schmocker, Timo Lang
```

```

00005 * Institution:    BFH Bern University of Applied Sciences
00006 * File:           common/touch/touch.c
00007 *
00008 * Version History:
00009 * Date            Autor Email      SHA      Changes
00010 * 2015-04-27      timolang@gmail.com 259d446 Added touch support to emulator. Implemented basic touch
           function.
00011 * 2015-05-02      timolang@gmail.com 3281616 Added some more touch functions. Improved pixy test. Drag the
           Image around!
00012 * 2015-05-17      timolang@gmail.com 2d46336 Improved comments in implementation of button, checkbox,
           numupdown, tft, touch and screen modules/submodules.
00013 * 2015-06-01      timolang@gmail.com 06227da Added calibrate screen (WIP). fixed bug in emulator drawing.
00014 * 2015-06-01      timolang@gmail.com eb573bc Finalized calibration. Fixed a bug in screen module.
00015 * 2015-06-06      timolang@gmail.com c06661d Fixed some outdated comments in source code. Documented Gui
           Module in docu.
00016 *
00017 *****
00018
00019 #include "touch.h"
00020 #include "ll_touch.h"
00021 #include "screen_calibrate.h"
00022 #include <stdio.h>
00023
00024 /* The idea is as follows:
00025  * The user can add "touch-areas" which basically represent a rectangles on the screen.
00026  * Once the user touches such a rectangle with the pen, we forward events to his provided callback.
00027  * Touch events are provided to us from the low level implementation via touch_add_raw_event().
00028  * We then need to check which touch areas are effected by that event
00029  */
00030
00031 /* Possible improvements:
00032  * Exchange pointer-list "areas" with a linked list. This would ensure that we can always accept new
           regions
00033  */
00034
00035 #define NUM_AREAS 50    //Number of Touch Areas we can manage
00036 TOUCH_AREA_STRUCT* areas[NUM_AREAS] = {NULL}; //list with pointers to all
           managed touch area's
00037
00038 volatile POINT_STRUCT pos; //the last touch point
00039 volatile TOUCH_STATE oldState = TOUCH_UP; //the last touch state
00040 volatile bool calibration = false; //whether or not we're currently calibrating
00041
00042 bool use_calibration = false; //Whether or not the current platform needs calibration and
           recalc of the values
00043
00044 //Calibration parameters (dummy values).
00045 int cal_xs = 10;
00046 int cal_dx = 100;
00047 int cal_ys = 10;
00048 int cal_dy = 100;
00049
00050
00051 void touch_set_calibration_values(int xs, int dx, int ys, int dy)
00052 {
00053     cal_xs = xs;
00054     cal_ys = ys;
00055     cal_dx = dx;
00056     cal_dy = dy;
00057 }
00058
00059
00060
00061 bool touch_init()
00062 {
00063     return ll_touch_init();
00064 }
00065
00066 void touch_set_value_convert_mode(bool uc)
00067 {
00068     use_calibration = uc;
00069 }
00070
00071
00072 bool touch_add_raw_event(uint16_t touchX, uint16_t touchY,
           TOUCH_STATE state)
00073 {
00074     //Update current and old position/state
00075     bool penDown = (state == TOUCH_DOWN);
00076     bool oldPenDown = (oldState == TOUCH_DOWN);
00077     oldState = state;
00078
00079     if (calibration) { //If in Calibration mode
00080         if (penDown) {
00081             pos.x = touchX;
00082             pos.y = touchY;

```

```

00083         } else {
00084             if (oldPenDown) { //Run only if we got at least one pen down
00085                 calibration = 0; //Calibration finish (Touch X and Y are the values from the
last measure, where the pen was down)
00086             }
00087         }
00088     }
00089     return true;
00090 }
00091
00092 //If we reach this point we're not in calibration mode and we need to process the event and call the
registered handlers..
00093
00094 if (use_calibration) { //the underlying touch hardware uses calibration
00095     //Calculate the real touch position out of the passed ones, and the calibration values
00096     pos.x = touchX = (((long) (DWIDTH - 2 * CCENTER) * 2 * (long) ((long)touchX -
cal_xs) / cal_dx + 1) >> 1) + CCENTER;
00097     pos.y = touchY = (((long) (DHEIGHT - 2 * CCENTER) * 2 * (long) ((long)touchY -
cal_ys) / cal_dy + 1) >> 1) + CCENTER;
00098 } else { //no conversion needed for the underlying hardware
00099     pos.x = touchX;
00100     pos.y = touchY;
00101 }
00102
00103 if (penDown) { //pen is down now
00104     //tft_draw_pixel(touchX,touchY,WHITE);
00105     if (!oldPenDown) { //pen wasn't down before (positive edge) => First Touch
00106         for (int z = 0; z < NUM_AREAS; z++) { // For every touch area
00107             //Check if pos is inside area
00108             if (areas[z] != NULL && touchX >= areas[z]->x1 && touchX <= areas[z]->x2 && touchY >=
areas[z]->y1 && touchY <= areas[z]->y2) {
00109                 areas[z]->flags = 1; //Save PenInside=1
00110
00111                 if (areas[z]->hookedActions & PEN_DOWN) { //The user wants to receive pen down
events
00112                     areas[z]->callback(areas[z], PEN_DOWN); //Send event to user callback
00113                 }
00114             }
00115         }
00116     } else { //Pen was down before => Second, Third event in row
00117         for (int z = 0; z < NUM_AREAS; z++) { // For every touch area
00118             if (areas[z] != NULL) {
00119                 //Check if pos is inside area
00120                 if (touchX >= areas[z]->x1 && touchX <= areas[z]->x2 && touchY >= areas[z]->y1 &&
touchY <= areas[z]->y2) {
00121                     if (areas[z]->flags == 0) { //Pen was not inside before (PenInside==0)
00122                         areas[z]->flags = 1; //Pen is inside now (PenInside=1)
00123
00124                     if (areas[z]->hookedActions & PEN_ENTER) { //The user wants to receive
pen enter events
00125                         areas[z]->callback(areas[z], PEN_ENTER);
00126                     }
00127                 }
00128                 } else if (areas[z]->flags) { //Pos not inside area, but it was before (PenInside==1)
00129                     areas[z]->flags = 0; //Pen is no longer inside (PenInside=0)
00130
00131                 if (areas[z]->hookedActions & PEN_LEAVE) { //The user wants to receive pen
leave events
00132                     areas[z]->callback(areas[z], PEN_LEAVE);
00133                 }
00134             }
00135         }
00136     }
00137 }
00138
00139 for (int z = 0; z < NUM_AREAS; z++) { // For every touch area
00140     if (areas[z] != NULL && (areas[z]->hookedActions & PEN_MOVE)) { //User want's to
receive pen move events
00141         //Check if pos is inside area
00142         if (touchX >= areas[z]->x1 && touchX <= areas[z]->x2 && touchY >= areas[z]->y1 && touchY <=
areas[z]->y2) {
00143             areas[z]->callback(areas[z], PEN_MOVE);
00144         }
00145     }
00146 }
00147 } else { //pen is not down now
00148     if (oldPenDown) { //but it was down before (negative edge)
00149         for (int z = 0; z < NUM_AREAS; z++) { // For every touch area
00150             //Check if pos is inside area
00151             if (areas[z] != NULL && touchX >= areas[z]->x1 && touchX <= areas[z]->x2 && touchY >= areas
[z]->y1 && touchY <= areas[z]->y2) {
00152                 areas[z]->flags = 0; //The pen is no longer inside (PenInside = 0);
00153
00154                 if (areas[z]->hookedActions & PEN_UP) { //user want's to receive pen up events
00155                     areas[z]->callback(areas[z], PEN_UP);
00156                 }
00157             }
00158         }
00159     }
00160 }

```

```

00158         }
00159     }
00160 }
00161
00162     return true;
00163 }
00164
00165 bool touch_have_empty(unsigned char num)
00166 {
00167     //go through pointer array and check for free spaces
00168     for (unsigned char i = 0; i < NUM_AREAS; i++) {
00169         if (areas[i] == NULL) {
00170             num--;    //a free space was found, we need one less
00171         }
00172     }
00173     if (num == 0) {
00174         return true;    //enough free spaces found
00175     }
00176 }
00177
00178     return false; //not enough free spaces found
00179 }
00180
00181 bool touch_register_area( TOUCH_AREA_STRUCT* area)
00182 {
00183     //go through pointer array and check for free space
00184     for (unsigned char i = 0; i < NUM_AREAS; i++) {
00185         if (areas[i] == NULL) { //free space found
00186             area->flags = 0; //we start with empty flags (PenInside=0)
00187             areas[i] = area; //save pointer into list
00188             return true;
00189         }
00190     }
00191
00192     return false; //no free space found
00193 }
00194
00195 void touch_unregister_area( TOUCH_AREA_STRUCT* area)
00196 {
00197     if (area == NULL) {
00198         return;
00199     }
00200
00201     //go through pointer array and find the area to remove
00202     for (unsigned char i = 0; i < NUM_AREAS; i++) {
00203         if (areas[i] == area) { //area found in pointer array at pos i
00204             areas[i] = NULL; //set pointer in list to NULL again
00205             break;
00206         }
00207     }
00208 }
00209
00210
00211 POINT_STRUCT touch_get_last_point()
00212 {
00213     return pos;
00214 }

```

7.87 common/touch/touch.h File Reference

```

#include <stdbool.h>
#include <stdint.h>

```


7.88 touch.h

```

00001 /*****
00002  * Project:      discoverpixy
00003  * Website:     https://github.com/t-moe/discoverpixy
00004  * Authors:     Aaron Schmocker, Timo Lang
00005  * Institution:  BFH Bern University of Applied Sciences
00006  * File:        common/touch/touch.h
00007  *
00008  * Version History:
00009  * Date      Autor Email      SHA      Changes
00010  * 2015-04-03 timolang@gmail.com 51089aa Refactored Project Structure for use with emulator
00011  * 2015-04-27 timolang@gmail.com 259d446 Added touch support to emulator. Implemented basic touch
00012  * 2015-04-27 timolang@gmail.com cf72baa Introduced a Screen (sub) module and divided app into multiple
00013  * 2015-05-02 timolang@gmail.com 3281616 Added some more touch functions. Improved pixy test. Drag the
00014  * 2015-05-11 timolang@gmail.com a175a2f Added doxygen docu for touch module
00015  * 2015-05-11 timolang@gmail.com 08d9fe0 More work on doxygen module structure
00016  * 2015-05-12 timolang@gmail.com 1402598 Added doxygen stuff for button module and some minor changes to
00017  * 2015-05-15 timolang@gmail.com 9a16865 Added doxygen comments to filesystem, checkbox, numupdown and
00018  * 2015-06-01 timolang@gmail.com 06227da Added calibrate screen (WIP). fixed bug in emulator drawing.
00019  * 2015-06-01 timolang@gmail.com eb573bc Finalized calibration. Fixed a bug in screen module.
00020  *
00021  *****/
00022
00023 #ifndef TOUCH_H
00024 #define TOUCH_H
00025
00026 #include<stdbool.h>
00027 #include<stdint.h>
00028
00029
00043 typedef enum {
00044     TOUCH_UP,
00045     TOUCH_DOWN
00046 } TOUCH_STATE ;
00047
00052 typedef enum {
00053     NONE = 0x00,
00054     PEN_DOWN = 0x01,
00055     PEN_UP = 0x02,
00056     PEN_ENTER = 0x04,
00057     PEN_LEAVE = 0x08,
00058     PEN_MOVE = 0x10
00059 } TOUCH_ACTION;
00060
00067 typedef void (*TOUCH_CALLBACK)(void* touchArea, TOUCH_ACTION triggeredAction);
00068
00072 typedef struct {
00073     TOUCH_ACTION hookedActions;
00074     uint16_t x1;
00075     uint16_t y1;
00076     uint16_t x2;
00077     uint16_t y2;
00078     TOUCH_CALLBACK callback;
00079     uint8_t flags;
00080 } TOUCH_AREA_STRUCT;
00081
00082
00086 typedef struct {
00087     uint16_t x;
00088     uint16_t y;
00089 } POINT_STRUCT;
00090
00096 bool touch_init();
00097
00107 bool touch_add_raw_event(uint16_t x, uint16_t y, TOUCH_STATE
state);
00108
00114 bool touch_have_empty(unsigned char num);
00115
00121 bool touch_register_area(TOUCH_AREA_STRUCT* area);
00122
00127 void touch_unregister_area(TOUCH_AREA_STRUCT* area);
00128
00133 POINT_STRUCT touch_get_last_point();
00134
00142 void touch_set_calibration_values(int xs, int dx, int ys, int dy);
00143
00144

```



```
00149 void touch_set_value_convert_mode (bool
      use_calibration);
00150
00153 #endif /* TOUCH_H */
```

