

## LS-CAT PGPMAC

Generated by Doxygen 1.8.2

Wed May 22 2013 15:30:30



# Contents

<b>1</b>	<b>The LS-CAT pgpmac Project</b>	<b>1</b>
<b>2</b>	<b>Namespace Index</b>	<b>5</b>
2.1	Namespace List . . . . .	5
<b>3</b>	<b>Data Structure Index</b>	<b>7</b>
3.1	Data Structures . . . . .	7
<b>4</b>	<b>File Index</b>	<b>9</b>
4.1	File List . . . . .	9
<b>5</b>	<b>Namespace Documentation</b>	<b>11</b>
5.1	iniParser Namespace Reference . . . . .	11
5.1.1	Variable Documentation . . . . .	11
5.1.1.1	ip . . . . .	11
5.2	mk_pgpmac_redis Namespace Reference . . . . .	11
5.2.1	Function Documentation . . . . .	12
5.2.1.1	active_simulation . . . . .	12
5.2.1.2	asis . . . . .	12
5.2.1.3	mk_active_init . . . . .	12
5.2.1.4	mk_home . . . . .	13
5.2.1.5	mk_inactive_init . . . . .	13
5.2.2	Variable Documentation . . . . .	13
5.2.2.1	b . . . . .	13
5.2.2.2	bi_list . . . . .	13
5.2.2.3	configs . . . . .	13
5.2.2.4	f . . . . .	14
5.2.2.5	fnc . . . . .	14
5.2.2.6	hard_ini . . . . .	14
5.2.2.7	hard_ini_fields . . . . .	14
5.2.2.8	head . . . . .	14
5.2.2.9	hi . . . . .	14
5.2.2.10	i . . . . .	14

5.2.2.11	motor_dict	14
5.2.2.12	motor_field_lists	14
5.2.2.13	motor_num	15
5.2.2.14	motor_presets	15
5.2.2.15	p	15
5.2.2.16	pi	15
5.2.2.17	plcc2_dict	15
5.2.2.18	plcc2_file	15
5.2.2.19	ppos	15
5.2.2.20	pref_ini	16
5.2.2.21	v	16
5.2.2.22	x	16
5.2.2.23	xlate	16
5.2.2.24	y	16
5.2.2.25	zoom_settings	16
<b>6</b>	<b>Data Structure Documentation</b>	<b>17</b>
6.1	iniParser.iniParser Class Reference	17
6.1.1	Detailed Description	17
6.1.2	Constructor & Destructor Documentation	18
6.1.2.1	__init__	18
6.1.3	Member Function Documentation	18
6.1.3.1	get	18
6.1.3.2	has_option	18
6.1.3.3	has_section	18
6.1.3.4	options	18
6.1.3.5	read	19
6.1.3.6	sections	19
6.1.4	Field Documentation	19
6.1.4.1	f	19
6.1.4.2	sd	19
6.2	Isevents_callbacks_struct Struct Reference	19
6.2.1	Detailed Description	20
6.2.2	Field Documentation	20
6.2.2.1	cb	20
6.2.2.2	next	20
6.3	Isevents_event_names_struct Struct Reference	20
6.3.1	Detailed Description	20
6.3.2	Field Documentation	20
6.3.2.1	cbl	20

6.3.2.2	event	20
6.3.2.3	next	21
6.4	Isevents_listener_struct Struct Reference	21
6.4.1	Detailed Description	21
6.4.2	Field Documentation	21
6.4.2.1	cb	21
6.4.2.2	next	21
6.4.2.3	raw_regexp	21
6.4.2.4	re	22
6.5	Isevents_queue_struct Struct Reference	22
6.5.1	Detailed Description	22
6.5.2	Field Documentation	22
6.5.2.1	evp	22
6.6	Islogging_queue_struct Struct Reference	22
6.6.1	Detailed Description	23
6.6.2	Field Documentation	23
6.6.2.1	lmsg	23
6.6.2.2	ltime	23
6.7	lspg_demandairrights_struct Struct Reference	23
6.7.1	Detailed Description	23
6.7.2	Field Documentation	23
6.7.2.1	cond	23
6.7.2.2	mutex	23
6.7.2.3	new_value_ready	23
6.8	lspg_getcenter_struct Struct Reference	24
6.8.1	Detailed Description	24
6.8.2	Field Documentation	24
6.8.2.1	cond	24
6.8.2.2	dax	25
6.8.2.3	dax_isnull	25
6.8.2.4	day	25
6.8.2.5	day_isnull	25
6.8.2.6	daz	25
6.8.2.7	daz_isnull	25
6.8.2.8	dcx	25
6.8.2.9	dcx_isnull	25
6.8.2.10	dcy	25
6.8.2.11	dcy_isnull	25
6.8.2.12	mutex	26
6.8.2.13	new_value_ready	26

6.8.2.14	no_rows_returned . . . . .	26
6.8.2.15	zoom . . . . .	26
6.8.2.16	zoom_isnull . . . . .	26
6.9	<b>lspg_getcurrentsampleid_struct Struct Reference</b> . . . . .	26
6.9.1	Detailed Description . . . . .	27
6.9.2	Field Documentation . . . . .	27
6.9.2.1	cond . . . . .	27
6.9.2.2	getcurrentsampleid . . . . .	27
6.9.2.3	getcurrentsampleid_isnull . . . . .	27
6.9.2.4	mutex . . . . .	27
6.9.2.5	new_value_ready . . . . .	27
6.9.2.6	no_rows_returned . . . . .	27
6.10	<b>lspg_lock_detector_struct Struct Reference</b> . . . . .	27
6.10.1	Detailed Description . . . . .	28
6.10.2	Field Documentation . . . . .	28
6.10.2.1	cond . . . . .	28
6.10.2.2	mutex . . . . .	28
6.10.2.3	new_value_ready . . . . .	28
6.11	<b>lspg_lock_diffractometer_struct Struct Reference</b> . . . . .	28
6.11.1	Detailed Description . . . . .	28
6.11.2	Field Documentation . . . . .	28
6.11.2.1	cond . . . . .	28
6.11.2.2	mutex . . . . .	28
6.11.2.3	new_value_ready . . . . .	29
6.12	<b>lspg_nexsample_struct Struct Reference</b> . . . . .	29
6.12.1	Detailed Description . . . . .	29
6.12.2	Field Documentation . . . . .	29
6.12.2.1	cond . . . . .	29
6.12.2.2	mutex . . . . .	29
6.12.2.3	new_value_ready . . . . .	29
6.12.2.4	nexsample . . . . .	30
6.12.2.5	nexsample_isnull . . . . .	30
6.12.2.6	no_rows_returned . . . . .	30
6.13	<b>lspg_nextshot_struct Struct Reference</b> . . . . .	30
6.13.1	Detailed Description . . . . .	33
6.13.2	Field Documentation . . . . .	33
6.13.2.1	active . . . . .	33
6.13.2.2	active2 . . . . .	33
6.13.2.3	active2_isnull . . . . .	33
6.13.2.4	active_isnull . . . . .	33

6.13.2.5	ax	33
6.13.2.6	ax2	33
6.13.2.7	ax2_isnull	33
6.13.2.8	ax_isnull	33
6.13.2.9	ay	34
6.13.2.10	ay2	34
6.13.2.11	ay2_isnull	34
6.13.2.12	ay_isnull	34
6.13.2.13	az	34
6.13.2.14	az2	34
6.13.2.15	az2_isnull	34
6.13.2.16	az_isnull	34
6.13.2.17	cond	34
6.13.2.18	cx	34
6.13.2.19	cx2	35
6.13.2.20	cx2_isnull	35
6.13.2.21	cx_isnull	35
6.13.2.22	cy	35
6.13.2.23	cy2	35
6.13.2.24	cy2_isnull	35
6.13.2.25	cy_isnull	35
6.13.2.26	dsdir	35
6.13.2.27	dsdir_isnull	35
6.13.2.28	dsdist	35
6.13.2.29	dsdist2	36
6.13.2.30	dsdist2_isnull	36
6.13.2.31	dsdist_isnull	36
6.13.2.32	dsexp	36
6.13.2.33	dsexp2	36
6.13.2.34	dsexp2_isnull	36
6.13.2.35	dsexp_isnull	36
6.13.2.36	dshpid	36
6.13.2.37	dshpid_isnull	36
6.13.2.38	dskappa	36
6.13.2.39	dskappa2	37
6.13.2.40	dskappa2_isnull	37
6.13.2.41	dskappa_isnull	37
6.13.2.42	dsnrg	37
6.13.2.43	dsnrg2	37
6.13.2.44	dsnrg2_isnull	37

6.13.2.45 dsnrg_isnull . . . . .	37
6.13.2.46 dsomega . . . . .	37
6.13.2.47 dsomega2 . . . . .	37
6.13.2.48 dsomega2_isnull . . . . .	37
6.13.2.49 dsomega_isnull . . . . .	37
6.13.2.50 dsoscaxis . . . . .	38
6.13.2.51 dsoscaxis2 . . . . .	38
6.13.2.52 dsoscaxis2_isnull . . . . .	38
6.13.2.53 dsoscaxis_isnull . . . . .	38
6.13.2.54 dsowidth . . . . .	38
6.13.2.55 dsowidth2 . . . . .	38
6.13.2.56 dsowidth2_isnull . . . . .	38
6.13.2.57 dsowidth_isnull . . . . .	38
6.13.2.58 dphi . . . . .	38
6.13.2.59 dphi2 . . . . .	38
6.13.2.60 dphi2_isnull . . . . .	39
6.13.2.61 dphi_isnull . . . . .	39
6.13.2.62 dpid . . . . .	39
6.13.2.63 dpid_isnull . . . . .	39
6.13.2.64 mutex . . . . .	39
6.13.2.65 new_value_ready . . . . .	39
6.13.2.66 no_rows_returned . . . . .	39
6.13.2.67 sfn . . . . .	39
6.13.2.68 sfn_isnull . . . . .	39
6.13.2.69 sindex . . . . .	39
6.13.2.70 sindex2 . . . . .	40
6.13.2.71 sindex2_isnull . . . . .	40
6.13.2.72 sindex_isnull . . . . .	40
6.13.2.73 skey . . . . .	40
6.13.2.74 skey_isnull . . . . .	40
6.13.2.75 sstart . . . . .	40
6.13.2.76 sstart2 . . . . .	40
6.13.2.77 sstart2_isnull . . . . .	40
6.13.2.78 sstart_isnull . . . . .	40
6.13.2.79 stype . . . . .	40
6.13.2.80 stype2 . . . . .	41
6.13.2.81 stype2_isnull . . . . .	41
6.13.2.82 stype_isnull . . . . .	41
6.14 lsgc_seq_run_prep_struct Struct Reference . . . . .	41
6.14.1 Detailed Description . . . . .	41

---

6.14.2 Field Documentation . . . . .	41
6.14.2.1 cond . . . . .	41
6.14.2.2 mutex . . . . .	41
6.14.2.3 new_value_ready . . . . .	41
6.15 lspg_starttransfer_struct Struct Reference . . . . .	42
6.15.1 Detailed Description . . . . .	42
6.15.2 Field Documentation . . . . .	42
6.15.2.1 cond . . . . .	42
6.15.2.2 mutex . . . . .	42
6.15.2.3 new_value_ready . . . . .	42
6.15.2.4 no_rows_returned . . . . .	42
6.15.2.5 starttransfer . . . . .	43
6.16 lspg_wait_for_detector_struct Struct Reference . . . . .	43
6.16.1 Detailed Description . . . . .	43
6.16.2 Field Documentation . . . . .	43
6.16.2.1 cond . . . . .	43
6.16.2.2 mutex . . . . .	43
6.16.2.3 new_value_ready . . . . .	43
6.17 lspg_waitcryo_struct Struct Reference . . . . .	43
6.17.1 Detailed Description . . . . .	44
6.17.2 Field Documentation . . . . .	44
6.17.2.1 cond . . . . .	44
6.17.2.2 mutex . . . . .	44
6.17.2.3 new_value_ready . . . . .	44
6.18 lspgQueryQueueStruct Struct Reference . . . . .	44
6.18.1 Detailed Description . . . . .	45
6.18.2 Field Documentation . . . . .	45
6.18.2.1 onResponse . . . . .	45
6.18.2.2 qs . . . . .	45
6.19 lspmacc_ascii_buffers_struct Struct Reference . . . . .	45
6.19.1 Detailed Description . . . . .	45
6.19.2 Field Documentation . . . . .	45
6.19.2.1 command_buf . . . . .	45
6.19.2.2 command_buf_cc . . . . .	45
6.19.2.3 command_str . . . . .	46
6.19.2.4 response_buf . . . . .	46
6.19.2.5 response_n . . . . .	46
6.19.2.6 response_str . . . . .	46
6.20 lspmacc_bi_struct Struct Reference . . . . .	46
6.20.1 Detailed Description . . . . .	46

---

6.20.2 Field Documentation . . . . .	47
6.20.2.1 changeEventOff . . . . .	47
6.20.2.2 changeEventOn . . . . .	47
6.20.2.3 first_time . . . . .	47
6.20.2.4 mask . . . . .	47
6.20.2.5 mutex . . . . .	47
6.20.2.6 position . . . . .	47
6.20.2.7 previous . . . . .	47
6.20.2.8 ptr . . . . .	47
6.21 lspmaccmd_queue_struct Struct Reference . . . . .	48
6.21.1 Detailed Description . . . . .	48
6.21.2 Field Documentation . . . . .	48
6.21.2.1 event . . . . .	48
6.21.2.2 no_reply . . . . .	48
6.21.2.3 onResponse . . . . .	48
6.21.2.4 pcmd . . . . .	48
6.21.2.5 time_sent . . . . .	49
6.22 lsmac_combined_move_struct Struct Reference . . . . .	49
6.22.1 Detailed Description . . . . .	49
6.22.2 Field Documentation . . . . .	49
6.22.2.1 axis . . . . .	49
6.22.2.2 coord_num . . . . .	49
6.22.2.3 Delta . . . . .	49
6.22.2.4 moveme . . . . .	49
6.23 lsmacd_pascal_queue_struct Struct Reference . . . . .	49
6.23.1 Detailed Description . . . . .	50
6.23.2 Field Documentation . . . . .	50
6.23.2.1 event . . . . .	50
6.23.2.2 pl . . . . .	50
6.24 lsmac_motor_struct Struct Reference . . . . .	50
6.24.1 Detailed Description . . . . .	52
6.24.2 Field Documentation . . . . .	52
6.24.2.1 active . . . . .	52
6.24.2.2 active_init . . . . .	52
6.24.2.3 actual_pos_cnts . . . . .	53
6.24.2.4 actual_pos_cnts_p . . . . .	53
6.24.2.5 axis . . . . .	53
6.24.2.6 command_sent . . . . .	53
6.24.2.7 cond . . . . .	53
6.24.2.8 coord_num . . . . .	53

6.24.2.9	dac_mvar	53
6.24.2.10	home	53
6.24.2.11	homing	53
6.24.2.12	in_position_band	54
6.24.2.13	inactive_init	54
6.24.2.14	jogAbs	54
6.24.2.15	lut	54
6.24.2.16	magic	54
6.24.2.17	max_accel	54
6.24.2.18	max_pos	54
6.24.2.19	max_speed	54
6.24.2.20	min_pos	54
6.24.2.21	motion_seen	55
6.24.2.22	motor_num	55
6.24.2.23	moveAbs	55
6.24.2.24	mutex	55
6.24.2.25	name	55
6.24.2.26	neg_limit_hit	55
6.24.2.27	neutral_pos	55
6.24.2.28	nlut	55
6.24.2.29	not_done	55
6.24.2.30	pos_limit_hit	56
6.24.2.31	position	56
6.24.2.32	pq	56
6.24.2.33	precision	56
6.24.2.34	printf_fmt	56
6.24.2.35	read	56
6.24.2.36	read_mask	56
6.24.2.37	read_ptr	56
6.24.2.38	redis_fmt	56
6.24.2.39	redis_position	57
6.24.2.40	reported_pg_position	57
6.24.2.41	reported_position	57
6.24.2.42	requested_pos cnts	57
6.24.2.43	requested_position	57
6.24.2.44	status1	57
6.24.2.45	status1_p	57
6.24.2.46	status2	57
6.24.2.47	status2_p	57
6.24.2.48	status_str	58

6.24.2.49 u2c . . . . .	58
6.24.2.50 unit . . . . .	58
6.24.2.51 update_resolution . . . . .	58
6.24.2.52 win . . . . .	58
6.24.2.53 write_fmt . . . . .	58
6.25 lsredis_obj_struct Struct Reference . . . . .	58
6.25.1 Detailed Description . . . . .	59
6.25.2 Field Documentation . . . . .	59
6.25.2.1 avalue . . . . .	59
6.25.2.2 bvalue . . . . .	59
6.25.2.3 cond . . . . .	59
6.25.2.4 cvalue . . . . .	59
6.25.2.5 dvalue . . . . .	60
6.25.2.6 events_name . . . . .	60
6.25.2.7 hits . . . . .	60
6.25.2.8 key . . . . .	60
6.25.2.9 lvalue . . . . .	60
6.25.2.10 mutex . . . . .	60
6.25.2.11 next . . . . .	60
6.25.2.12 valid . . . . .	60
6.25.2.13 value . . . . .	60
6.25.2.14 value_length . . . . .	61
6.25.2.15 wait_for_me . . . . .	61
6.26 lsredis_preset_list_struct Struct Reference . . . . .	61
6.26.1 Detailed Description . . . . .	61
6.26.2 Field Documentation . . . . .	61
6.26.2.1 index . . . . .	61
6.26.2.2 key . . . . .	61
6.26.2.3 name . . . . .	61
6.26.2.4 next . . . . .	61
6.26.2.5 position . . . . .	62
6.27 lstimber_list_struct Struct Reference . . . . .	62
6.27.1 Detailed Description . . . . .	62
6.27.2 Field Documentation . . . . .	62
6.27.2.1 delay_nsecs . . . . .	62
6.27.2.2 delay_secs . . . . .	63
6.27.2.3 event . . . . .	63
6.27.2.4 init_nsecs . . . . .	63
6.27.2.5 init_secs . . . . .	63
6.27.2.6 last_nsecs . . . . .	63

---

6.27.2.7	last_secs . . . . .	63
6.27.2.8	ncalls . . . . .	63
6.27.2.9	next_nsecs . . . . .	63
6.27.2.10	next_secs . . . . .	63
6.27.2.11	shots . . . . .	64
6.28	md2cmds_cmd_kv_struct Struct Reference . . . . .	64
6.28.1	Detailed Description . . . . .	64
6.28.2	Field Documentation . . . . .	64
6.28.2.1	k . . . . .	64
6.28.2.2	v . . . . .	64
6.29	md2StatusStruct Struct Reference . . . . .	64
6.29.1	Detailed Description . . . . .	66
6.29.2	Field Documentation . . . . .	66
6.29.2.1	acc11c_1 . . . . .	66
6.29.2.2	acc11c_2 . . . . .	66
6.29.2.3	acc11c_3 . . . . .	66
6.29.2.4	acc11c_5 . . . . .	66
6.29.2.5	acc11c_6 . . . . .	66
6.29.2.6	alignx_act_pos . . . . .	66
6.29.2.7	alignx_status_1 . . . . .	66
6.29.2.8	alignx_status_2 . . . . .	66
6.29.2.9	aligny_act_pos . . . . .	66
6.29.2.10	aligny_status_1 . . . . .	67
6.29.2.11	aligny_status_2 . . . . .	67
6.29.2.12	alignz_act_pos . . . . .	67
6.29.2.13	alignz_status_1 . . . . .	67
6.29.2.14	alignz_status_2 . . . . .	67
6.29.2.15	analyzer_act_pos . . . . .	67
6.29.2.16	analyzer_status_1 . . . . .	67
6.29.2.17	analyzer_status_2 . . . . .	67
6.29.2.18	aperturey_act_pos . . . . .	67
6.29.2.19	aperturey_status_1 . . . . .	67
6.29.2.20	aperturey_status_2 . . . . .	67
6.29.2.21	aperturez_act_pos . . . . .	67
6.29.2.22	aperturez_status_1 . . . . .	68
6.29.2.23	aperturez_status_2 . . . . .	68
6.29.2.24	back_dac . . . . .	68
6.29.2.25	capy_act_pos . . . . .	68
6.29.2.26	capy_status_1 . . . . .	68
6.29.2.27	capy_status_2 . . . . .	68

6.29.2.28 capz_act_pos . . . . .	68
6.29.2.29 capz_status_1 . . . . .	68
6.29.2.30 capz_status_2 . . . . .	68
6.29.2.31 centerx_act_pos . . . . .	68
6.29.2.32 centerx_status_1 . . . . .	68
6.29.2.33 centerx_status_2 . . . . .	68
6.29.2.34 centery_act_pos . . . . .	69
6.29.2.35 centery_status_1 . . . . .	69
6.29.2.36 centery_status_2 . . . . .	69
6.29.2.37 dummy1 . . . . .	69
6.29.2.38 dummy2 . . . . .	69
6.29.2.39 dummy3 . . . . .	69
6.29.2.40 dummy4 . . . . .	69
6.29.2.41 dummy5 . . . . .	69
6.29.2.42 dummy6 . . . . .	69
6.29.2.43 dummy7 . . . . .	69
6.29.2.44 dummy8 . . . . .	69
6.29.2.45 dummy9 . . . . .	69
6.29.2.46 dummyA . . . . .	70
6.29.2.47 dummyB . . . . .	70
6.29.2.48 front_dac . . . . .	70
6.29.2.49 fs_has_opened . . . . .	70
6.29.2.50 fs_has_opened_globally . . . . .	70
6.29.2.51 fs_is_open . . . . .	70
6.29.2.52 kappa_act_pos . . . . .	70
6.29.2.53 kappa_status_1 . . . . .	70
6.29.2.54 kappa_status_2 . . . . .	70
6.29.2.55 moving_flags . . . . .	70
6.29.2.56 number_passes . . . . .	70
6.29.2.57 omega_act_pos . . . . .	70
6.29.2.58 omega_status_1 . . . . .	71
6.29.2.59 omega_status_2 . . . . .	71
6.29.2.60 phi_act_pos . . . . .	71
6.29.2.61 phi_status_1 . . . . .	71
6.29.2.62 phi_status_2 . . . . .	71
6.29.2.63 phiscan . . . . .	71
6.29.2.64 scint_act_pos . . . . .	71
6.29.2.65 scint_piezo . . . . .	71
6.29.2.66 scint_status_1 . . . . .	71
6.29.2.67 scint_status_2 . . . . .	71

6.29.2.68 zoom_act_pos . . . . .	71
6.29.2.69 zoom_status_1 . . . . .	71
6.29.2.70 zoom_status_2 . . . . .	72
6.30 tagEthernetCmd Struct Reference . . . . .	72
6.30.1 Detailed Description . . . . .	72
6.30.2 Field Documentation . . . . .	72
6.30.2.1 bData . . . . .	72
6.30.2.2 Request . . . . .	72
6.30.2.3 RequestType . . . . .	73
6.30.2.4 wlIndex . . . . .	73
6.30.2.5 wLength . . . . .	73
6.30.2.6 wValue . . . . .	73
<b>7 File Documentation</b> . . . . .	<b>75</b>
7.1 iniParser.py File Reference . . . . .	75
7.2 lsevents.c File Reference . . . . .	75
7.2.1 Detailed Description . . . . .	77
7.2.2 Macro Definition Documentation . . . . .	77
7.2.2.1 LSEVENTS_QUEUE_LENGTH . . . . .	77
7.2.3 Typedef Documentation . . . . .	77
7.2.3.1 lsevents_callbacks_t . . . . .	77
7.2.3.2 lsevents_event_names_t . . . . .	77
7.2.3.3 lsevents_listener_t . . . . .	77
7.2.3.4 lsevents_queue_t . . . . .	77
7.2.4 Function Documentation . . . . .	78
7.2.4.1 lsevents_add_listener . . . . .	78
7.2.4.2 lsevents_init . . . . .	78
7.2.4.3 lsevents_preregister_event . . . . .	79
7.2.4.4 lsevents_register_event . . . . .	79
7.2.4.5 lsevents_remove_listener . . . . .	80
7.2.4.6 lsevents_run . . . . .	81
7.2.4.7 lsevents_send_event . . . . .	81
7.2.4.8 lsevents_worker . . . . .	82
7.2.5 Variable Documentation . . . . .	83
7.2.5.1 lsevents_event_name_ht . . . . .	83
7.2.5.2 lsevents_event_names . . . . .	83
7.2.5.3 lsevents_listener_mutex . . . . .	83
7.2.5.4 lsevents_listeners_p . . . . .	83
7.2.5.5 lsevents_max_events . . . . .	83
7.2.5.6 lsevents_n_events . . . . .	83

7.2.5.7	Isevents_queue . . . . .	83
7.2.5.8	Isevents_queue_cond . . . . .	83
7.2.5.9	Isevents_queue_mutex . . . . .	83
7.2.5.10	Isevents_queue_off . . . . .	84
7.2.5.11	Isevents_queue_on . . . . .	84
7.2.5.12	Isevents_thread . . . . .	84
7.3	Islogging.c File Reference . . . . .	84
7.3.1	Detailed Description . . . . .	85
7.3.2	Macro Definition Documentation . . . . .	85
7.3.2.1	LSLOGGING_FILE_NAME . . . . .	85
7.3.2.2	LSLOGGING_MSG_LENGTH . . . . .	86
7.3.2.3	LSLOGGING_QUEUE_LENGTH . . . . .	86
7.3.3	Typedef Documentation . . . . .	86
7.3.3.1	Islogging_queue_t . . . . .	86
7.3.4	Function Documentation . . . . .	86
7.3.4.1	Islogging_event_cb . . . . .	86
7.3.4.2	Islogging_init . . . . .	86
7.3.4.3	Islogging_log_message . . . . .	86
7.3.4.4	Islogging_run . . . . .	87
7.3.4.5	Islogging_worker . . . . .	87
7.3.5	Variable Documentation . . . . .	88
7.3.5.1	Islogging_cond . . . . .	88
7.3.5.2	Islogging_file . . . . .	88
7.3.5.3	Islogging_mutex . . . . .	88
7.3.5.4	Islogging_off . . . . .	88
7.3.5.5	Islogging_on . . . . .	88
7.3.5.6	Islogging_queue . . . . .	88
7.3.5.7	Islogging_thread . . . . .	88
7.4	Isapg.c File Reference . . . . .	89
7.4.1	Detailed Description . . . . .	94
7.4.2	Macro Definition Documentation . . . . .	94
7.4.2.1	LS_PG_QUERY_QUEUE_LENGTH . . . . .	94
7.4.2.2	LS_PG_STATE_IDLE . . . . .	94
7.4.2.3	LS_PG_STATE_INIT . . . . .	94
7.4.2.4	LS_PG_STATE_INIT_POLL . . . . .	94
7.4.2.5	LS_PG_STATE_RECV . . . . .	95
7.4.2.6	LS_PG_STATE_RESET . . . . .	95
7.4.2.7	LS_PG_STATE_RESET_POLL . . . . .	95
7.4.2.8	LS_PG_STATE_SEND . . . . .	95
7.4.2.9	LS_PG_STATE_SEND_FLUSH . . . . .	95

7.4.3	Typedef Documentation . . . . .	95
7.4.3.1	lspg_lock_detector_t . . . . .	95
7.4.3.2	lspg_lock_diffractometer_t . . . . .	95
7.4.3.3	lspg_seq_run_prep_t . . . . .	95
7.4.3.4	lspg_wait_for_detector_t . . . . .	95
7.4.4	Function Documentation . . . . .	95
7.4.4.1	lspg_allkvs_cb . . . . .	95
7.4.4.2	lspg_array2ptrs . . . . .	96
7.4.4.3	lspg_check_preset_in_position_cb . . . . .	97
7.4.4.4	lspg_cmd_cb . . . . .	98
7.4.4.5	lspg_demandairrights_all . . . . .	98
7.4.4.6	lspg_demandairrights_call . . . . .	98
7.4.4.7	lspg_demandairrights_cb . . . . .	99
7.4.4.8	lspg_demandairrights_init . . . . .	99
7.4.4.9	lspg_demandairrights_wait . . . . .	99
7.4.4.10	lspg_flush . . . . .	99
7.4.4.11	lspg_getcenter_all . . . . .	100
7.4.4.12	lspg_getcenter_call . . . . .	100
7.4.4.13	lspg_getcenter_cb . . . . .	100
7.4.4.14	lspg_getcenter_done . . . . .	101
7.4.4.15	lspg_getcenter_init . . . . .	101
7.4.4.16	lspg_getcenter_wait . . . . .	101
7.4.4.17	lspg_getcurrentsampleid_call . . . . .	102
7.4.4.18	lspg_getcurrentsampleid_cb . . . . .	102
7.4.4.19	lspg_getcurrentsampleid_init . . . . .	102
7.4.4.20	lspg_getcurrentsampleid_read . . . . .	103
7.4.4.21	lspg_getcurrentsampleid_wait_for_id . . . . .	103
7.4.4.22	lspg_init . . . . .	103
7.4.4.23	lspg_lock_detector_all . . . . .	104
7.4.4.24	lspg_lock_detector_call . . . . .	104
7.4.4.25	lspg_lock_detector_cb . . . . .	104
7.4.4.26	lspg_lock_detector_done . . . . .	104
7.4.4.27	lspg_lock_detector_init . . . . .	104
7.4.4.28	lspg_lock_detector_wait . . . . .	105
7.4.4.29	lspg_lock_diffractometer_all . . . . .	105
7.4.4.30	lspg_lock_diffractometer_call . . . . .	105
7.4.4.31	lspg_lock_diffractometer_cb . . . . .	105
7.4.4.32	lspg_lock_diffractometer_done . . . . .	106
7.4.4.33	lspg_lock_diffractometer_init . . . . .	106
7.4.4.34	lspg_lock_diffractometer_wait . . . . .	106

7.4.4.35	lspg_next_state . . . . .	106
7.4.4.36	lspg_nextaction_cb . . . . .	107
7.4.4.37	lspg_nexterrors_cb . . . . .	107
7.4.4.38	lspg_nexsample_all . . . . .	108
7.4.4.39	lspg_nexsample_call . . . . .	108
7.4.4.40	lspg_nexsample_cb . . . . .	108
7.4.4.41	lspg_nexsample_done . . . . .	109
7.4.4.42	lspg_nexsample_init . . . . .	109
7.4.4.43	lspg_nexsample_wait . . . . .	109
7.4.4.44	lspg_nextshot_call . . . . .	110
7.4.4.45	lspg_nextshot_cb . . . . .	110
7.4.4.46	lspg_nextshot_done . . . . .	114
7.4.4.47	lspg_nextshot_init . . . . .	114
7.4.4.48	lspg_nextshot_wait . . . . .	114
7.4.4.49	lspg_notice_processor . . . . .	114
7.4.4.50	lspg_pg_connect . . . . .	114
7.4.4.51	lspg_pg_service . . . . .	116
7.4.4.52	lspg_preset_changed_cb . . . . .	117
7.4.4.53	lspg_query_next . . . . .	118
7.4.4.54	lspg_query_push . . . . .	118
7.4.4.55	lspg_query_reply_next . . . . .	119
7.4.4.56	lspg_query_reply_peek . . . . .	119
7.4.4.57	lspg_quitting_cb . . . . .	119
7.4.4.58	lspg_receive . . . . .	119
7.4.4.59	lspg_run . . . . .	120
7.4.4.60	lspg_sample_detector_cb . . . . .	121
7.4.4.61	lspg_send_next_query . . . . .	121
7.4.4.62	lspg_seq_run_prep_all . . . . .	122
7.4.4.63	lspg_seq_run_prep_call . . . . .	122
7.4.4.64	lspg_seq_run_prep_cb . . . . .	122
7.4.4.65	lspg_seq_run_prep_done . . . . .	123
7.4.4.66	lspg_seq_run_prep_init . . . . .	123
7.4.4.67	lspg_seq_run_prep_wait . . . . .	123
7.4.4.68	lspg_set_scale_cb . . . . .	123
7.4.4.69	lspg_sig_service . . . . .	124
7.4.4.70	lspg_starttransfer_all . . . . .	124
7.4.4.71	lspg_starttransfer_call . . . . .	124
7.4.4.72	lspg_starttransfer_cb . . . . .	125
7.4.4.73	lspg_starttransfer_done . . . . .	125
7.4.4.74	lspg_starttransfer_init . . . . .	125

---

7.4.4.75	lspg_starttransfer_wait . . . . .	125
7.4.4.76	lspg_unset_current_preset_moving_cb . . . . .	126
7.4.4.77	lspg_update_kvs_cb . . . . .	126
7.4.4.78	lspg_wait_for_detector_all . . . . .	127
7.4.4.79	lspg_wait_for_detector_call . . . . .	127
7.4.4.80	lspg_wait_for_detector_cb . . . . .	127
7.4.4.81	lspg_wait_for_detector_done . . . . .	128
7.4.4.82	lspg_wait_for_detector_init . . . . .	128
7.4.4.83	lspg_wait_for_detector_wait . . . . .	128
7.4.4.84	lspg_waitcryo_all . . . . .	128
7.4.4.85	lspg_waitcryo_cb . . . . .	129
7.4.4.86	lspg_waitcryo_init . . . . .	129
7.4.4.87	lspg_worker . . . . .	129
7.4.5	Variable Documentation . . . . .	130
7.4.5.1	ls_pg_state . . . . .	130
7.4.5.2	lspg_connectPoll_response . . . . .	130
7.4.5.3	lspg_demandairrights . . . . .	130
7.4.5.4	lspg_getcenter . . . . .	130
7.4.5.5	lspg_getcurrentsampleid . . . . .	130
7.4.5.6	lspg_lock_detector . . . . .	130
7.4.5.7	lspg_lock_diffractometer . . . . .	131
7.4.5.8	lspg_nexsample . . . . .	131
7.4.5.9	lspg_nextshot . . . . .	131
7.4.5.10	lspg_query_queue . . . . .	131
7.4.5.11	lspg_query_queue_off . . . . .	131
7.4.5.12	lspg_query_queue_on . . . . .	131
7.4.5.13	lspg_query_queue_reply . . . . .	131
7.4.5.14	lspg_queue_cond . . . . .	131
7.4.5.15	lspg_queue_mutex . . . . .	131
7.4.5.16	lspg_resetPoll_response . . . . .	132
7.4.5.17	lspg_seq_run_prep . . . . .	132
7.4.5.18	lspg_starttransfer . . . . .	132
7.4.5.19	lspg_thread . . . . .	132
7.4.5.20	lspg_wait_for_detector . . . . .	132
7.4.5.21	lspg_waitcryo . . . . .	132
7.4.5.22	lspgfd . . . . .	132
7.4.5.23	now . . . . .	132
7.4.5.24	q . . . . .	132
7.5	Ispmac.c File Reference . . . . .	133
7.5.1	Detailed Description . . . . .	141

7.5.2 Macro Definition Documentation . . . . .	142
7.5.2.1 LS_PMAC_STATE_CR . . . . .	142
7.5.2.2 LS_PMAC_STATE_DETACHED . . . . .	142
7.5.2.3 LS_PMAC_STATE_GB . . . . .	142
7.5.2.4 LS_PMAC_STATE_GMR . . . . .	142
7.5.2.5 LS_PMAC_STATE_IDLE . . . . .	143
7.5.2.6 LS_PMAC_STATE_RESET . . . . .	143
7.5.2.7 LS_PMAC_STATE_RR . . . . .	143
7.5.2.8 LS_PMAC_STATE_SC . . . . .	143
7.5.2.9 LS_PMAC_STATE_WACK . . . . .	143
7.5.2.10 LS_PMAC_STATE_WACK_CC . . . . .	143
7.5.2.11 LS_PMAC_STATE_WACK_NFR . . . . .	143
7.5.2.12 LS_PMAC_STATE_WACK_RR . . . . .	143
7.5.2.13 LS_PMAC_STATE_WCR . . . . .	143
7.5.2.14 LS_PMAC_STATE_WGB . . . . .	143
7.5.2.15 LSPMAC_DPASCII_QUEUE_LENGTH . . . . .	143
7.5.2.16 LSPMAC_MAX_MOTORS . . . . .	143
7.5.2.17 LSPMAC_PRESET_REGEX . . . . .	144
7.5.2.18 PMAC_CMD_QUEUE_LENGTH . . . . .	144
7.5.2.19 pmac_cmd_size . . . . .	144
7.5.2.20 PMAC_MIN_CMD_TIME . . . . .	144
7.5.2.21 PMACPORT . . . . .	144
7.5.2.22 VR_CTRL_RESPONSE . . . . .	144
7.5.2.23 VR_DOWNLOAD . . . . .	144
7.5.2.24 VR_FWDOWNLOAD . . . . .	144
7.5.2.25 VR_IPADDRESS . . . . .	144
7.5.2.26 VR_PMAC_FLUSH . . . . .	144
7.5.2.27 VR_PMAC_GETBUFFER . . . . .	144
7.5.2.28 VR_PMAC_GETLINE . . . . .	145
7.5.2.29 VR_PMAC_GETMEM . . . . .	145
7.5.2.30 VR_PMAC_GETRESPONSE . . . . .	145
7.5.2.31 VR_PMAC_PORT . . . . .	145
7.5.2.32 VR_PMAC_READREADY . . . . .	145
7.5.2.33 VR_PMAC_SENDCTRLCHAR . . . . .	145
7.5.2.34 VR_PMAC_SENDLINE . . . . .	145
7.5.2.35 VR_PMAC_SETBIT . . . . .	145
7.5.2.36 VR_PMAC_SETBITS . . . . .	145
7.5.2.37 VR_PMAC_SETMEM . . . . .	145
7.5.2.38 VR_PMAC_WRITEBUFFER . . . . .	145
7.5.2.39 VR_PMAC_WRITEERROR . . . . .	145

7.5.2.40	VR_UPLOAD	146
7.5.3	Typedef Documentation	146
7.5.3.1	lspmac_ascii_buffers_t	146
7.5.3.2	lspmac_combined_move_t	146
7.5.3.3	lspmac_dpascii_queue_t	146
7.5.3.4	md2_status_t	146
7.5.4	Function Documentation	146
7.5.4.1	_lspmac_motor_init	146
7.5.4.2	cleanstr	147
7.5.4.3	hex_dump	147
7.5.4.4	IsConnect	148
7.5.4.5	lspmac_abort	149
7.5.4.6	lspmac_asciiCmdCB	149
7.5.4.7	lspmac_backLight_down_cb	149
7.5.4.8	lspmac_backLight_up_cb	149
7.5.4.9	lspmac_bi_init	150
7.5.4.10	lspmac_blight_lut_setup	150
7.5.4.11	lspmac_bo_init	151
7.5.4.12	lspmac_bo_read	151
7.5.4.13	lspmac_command_done_cb	152
7.5.4.14	lspmac_cryoSwitchChanged_cb	152
7.5.4.15	lspmac_dac_init	152
7.5.4.16	lspmac_dac_read	153
7.5.4.17	lspmac_Error	153
7.5.4.18	lspmac_est_move_time	154
7.5.4.19	lspmac_est_move_time_wait	159
7.5.4.20	lspmac_find_motor_by_name	160
7.5.4.21	lspmac_flight_lut_setup	160
7.5.4.22	lspmac_fscint_lut_setup	161
7.5.4.23	lspmac_fshut_init	161
7.5.4.24	lspmac_full_card_reset_cb	161
7.5.4.25	lspmac_get_ascii	162
7.5.4.26	lspmac_get_ascii_cb	162
7.5.4.27	lspmac_get_status	163
7.5.4.28	lspmac_get_status_cb	164
7.5.4.29	lspmac_GetAllIVars	167
7.5.4.30	lspmac_GetAllIVarsCB	167
7.5.4.31	lspmac_GetAllMVars	167
7.5.4.32	lspmac_GetAllMVarsCB	167
7.5.4.33	lspmac_getBIPosition	168

7.5.4.34	Ispmac_Getmem . . . . .	168
7.5.4.35	Ispmac_GetmemReplyCB . . . . .	168
7.5.4.36	Ispmac_getPosition . . . . .	169
7.5.4.37	Ispmac_GetShortReplyCB . . . . .	169
7.5.4.38	Ispmac_home1_queue . . . . .	169
7.5.4.39	Ispmac_home2_queue . . . . .	171
7.5.4.40	Ispmac_init . . . . .	171
7.5.4.41	Ispmac_jogabs_queue . . . . .	175
7.5.4.42	Ispmac_light_zoom_cb . . . . .	175
7.5.4.43	Ispmac_lut . . . . .	176
7.5.4.44	Ispmac_more_ascii_cb . . . . .	177
7.5.4.45	Ispmac_motor_init . . . . .	177
7.5.4.46	Ispmac_move_or_jog_abs_queue . . . . .	178
7.5.4.47	Ispmac_move_or_jog_preset_queue . . . . .	181
7.5.4.48	Ispmac_move_preset_queue . . . . .	181
7.5.4.49	Ispmac_moveabs_blight_factor_queue . . . . .	182
7.5.4.50	Ispmac_moveabs_bo_queue . . . . .	182
7.5.4.51	Ispmac_moveabs_flight_factor_queue . . . . .	183
7.5.4.52	Ispmac_moveabs_frontlight_oo_queue . . . . .	183
7.5.4.53	Ispmac_moveabs_fshut_queue . . . . .	183
7.5.4.54	Ispmac_moveabs_queue . . . . .	184
7.5.4.55	Ispmac_moveabs_timed_queue . . . . .	184
7.5.4.56	Ispmac_moveabs_wait . . . . .	185
7.5.4.57	Ispmac_movedac_queue . . . . .	186
7.5.4.58	Ispmac_movezoom_queue . . . . .	187
7.5.4.59	Ispmac_next_state . . . . .	188
7.5.4.60	Ispmac_pmacmotor_read . . . . .	189
7.5.4.61	Ispmac_pop_queue . . . . .	193
7.5.4.62	Ispmac_pop_reply . . . . .	193
7.5.4.63	Ispmac_push_queue . . . . .	193
7.5.4.64	Ispmac_quitting_cb . . . . .	194
7.5.4.65	Ispmac_request_control_response_cb . . . . .	194
7.5.4.66	Ispmac_Reset . . . . .	194
7.5.4.67	Ispmac_reset_queue . . . . .	195
7.5.4.68	Ispmac_rlut . . . . .	195
7.5.4.69	Ispmac_run . . . . .	196
7.5.4.70	Ispmac_scint_dried_cb . . . . .	197
7.5.4.71	Ispmac_scint_maybe_move_sample_cb . . . . .	198
7.5.4.72	Ispmac_scint_maybe_return_sample_cb . . . . .	198
7.5.4.73	Ispmac_scint_maybe_turn_off_dryer_cb . . . . .	199

7.5.4.74	Ispmac_scint_maybe_turn_on_dryer_cb	199
7.5.4.75	Ispmac_send_command	200
7.5.4.76	Ispmac_sendcmd	200
7.5.4.77	Ispmac_sendcmd_nocb	201
7.5.4.78	Ispmac_SendControlReplyPrintCB	201
7.5.4.79	Ispmac_Service	202
7.5.4.80	Ispmac_set_motion_flags	204
7.5.4.81	Ispmac_shutter_read	205
7.5.4.82	Ispmac_SockFlush	206
7.5.4.83	Ispmac_SockGetmem	206
7.5.4.84	Ispmac_SockSendControlCharPrint	207
7.5.4.85	Ispmac_SockSendDPControlChar	207
7.5.4.86	Ispmac_SockSendDPControlCharCB	207
7.5.4.87	Ispmac_SockSendDPLine	207
7.5.4.88	Ispmac_SockSendDPqueue	208
7.5.4.89	Ispmac_SockSendline	208
7.5.4.90	Ispmac_SockSendline_nr	209
7.5.4.91	Ispmac_soft_motor_init	209
7.5.4.92	Ispmac_soft_motor_read	209
7.5.4.93	Ispmac_test_preset	210
7.5.4.94	Ispmac_video_rotate	210
7.5.4.95	Ispmac_worker	210
7.5.4.96	Ispmac_zoom_lut_setup	211
7.5.5	Variable Documentation	212
7.5.5.1	alignx	212
7.5.5.2	aligny	212
7.5.5.3	alignz	212
7.5.5.4	anal	212
7.5.5.5	apery	212
7.5.5.6	aperz	212
7.5.5.7	arm_parked	212
7.5.5.8	blight	212
7.5.5.9	blight_down	213
7.5.5.10	blight_f	213
7.5.5.11	blight_ud	213
7.5.5.12	blight_up	213
7.5.5.13	capy	213
7.5.5.14	capz	213
7.5.5.15	cenx	213
7.5.5.16	ceny	213

7.5.5.17	cr_cmd . . . . .	213
7.5.5.18	cryo . . . . .	214
7.5.5.19	cryo_back . . . . .	214
7.5.5.20	cryo_switch . . . . .	214
7.5.5.21	dbmem . . . . .	214
7.5.5.22	dbmemln . . . . .	214
7.5.5.23	dryer . . . . .	214
7.5.5.24	etel_init_ok . . . . .	214
7.5.5.25	etel_on . . . . .	214
7.5.5.26	etel_ready . . . . .	214
7.5.5.27	ethCmdOff . . . . .	215
7.5.5.28	ethCmdOn . . . . .	215
7.5.5.29	ethCmdQueue . . . . .	215
7.5.5.30	ethCmdReply . . . . .	215
7.5.5.31	flight . . . . .	215
7.5.5.32	flight_f . . . . .	215
7.5.5.33	flight_oo . . . . .	215
7.5.5.34	fluo . . . . .	215
7.5.5.35	fluor_back . . . . .	215
7.5.5.36	fscint . . . . .	216
7.5.5.37	fshut . . . . .	216
7.5.5.38	gb_cmd . . . . .	216
7.5.5.39	getivars . . . . .	216
7.5.5.40	getmvars . . . . .	216
7.5.5.41	hp_air . . . . .	216
7.5.5.42	kappa . . . . .	216
7.5.5.43	lp_air . . . . .	216
7.5.5.44	ls_pmac_state . . . . .	216
7.5.5.45	lspmac_ascii_buffers . . . . .	216
7.5.5.46	lspmac_ascii_buffers_mutex . . . . .	217
7.5.5.47	lspmac_ascii_busy . . . . .	217
7.5.5.48	lspmac_ascii_mutex . . . . .	217
7.5.5.49	lspmac_bis . . . . .	217
7.5.5.50	lspmac_control_char . . . . .	217
7.5.5.51	lspmac_dpascii_off . . . . .	217
7.5.5.52	lspmac_dpascii_on . . . . .	217
7.5.5.53	lspmac_dpascii_queue . . . . .	217
7.5.5.54	lspmac_motors . . . . .	217
7.5.5.55	lspmac_moving_cond . . . . .	217
7.5.5.56	lspmac_moving_flags . . . . .	218

7.5.5.57	<a href="#">lspmac_moving_mutex</a>	218
7.5.5.58	<a href="#">lspmac_nbis</a>	218
7.5.5.59	<a href="#">lspmac_nmotors</a>	218
7.5.5.60	<a href="#">lspmac_running</a>	218
7.5.5.61	<a href="#">lspmac_shutter_cond</a>	218
7.5.5.62	<a href="#">lspmac_shutter_has_opened</a>	218
7.5.5.63	<a href="#">lspmac_shutter_mutex</a>	218
7.5.5.64	<a href="#">lspmac_shutter_state</a>	218
7.5.5.65	<a href="#">lspmac_status_last_time</a>	219
7.5.5.66	<a href="#">lspmac_status_time</a>	219
7.5.5.67	<a href="#">md2_status</a>	219
7.5.5.68	<a href="#">md2_status_mutex</a>	219
7.5.5.69	<a href="#">minikappa_ok</a>	219
7.5.5.70	<a href="#">motors_ht</a>	219
7.5.5.71	<a href="#">now</a>	219
7.5.5.72	<a href="#">omega</a>	219
7.5.5.73	<a href="#">omega_zero_search</a>	219
7.5.5.74	<a href="#">omega_zero_time</a>	220
7.5.5.75	<a href="#">omega_zero_velocity</a>	220
7.5.5.76	<a href="#">phi</a>	220
7.5.5.77	<a href="#">pmac_error_strs</a>	220
7.5.5.78	<a href="#">pmac_queue_cond</a>	220
7.5.5.79	<a href="#">pmac_queue_mutex</a>	220
7.5.5.80	<a href="#">pmac_thread</a>	221
7.5.5.81	<a href="#">pmacfd</a>	221
7.5.5.82	<a href="#">rr_cmd</a>	221
7.5.5.83	<a href="#">sample_detected</a>	221
7.5.5.84	<a href="#">scint</a>	221
7.5.5.85	<a href="#">shutter_open</a>	221
7.5.5.86	<a href="#">smart_mag_err</a>	221
7.5.5.87	<a href="#">smart_mag_off</a>	221
7.5.5.88	<a href="#">smart_mag_on</a>	221
7.5.5.89	<a href="#">smart_mag_oo</a>	222
7.5.5.90	<a href="#">zoom</a>	222
7.6	<a href="#">lsredis.c File Reference</a>	222
7.6.1	<a href="#">Detailed Description</a>	224
7.6.2	<a href="#">Typedef Documentation</a>	225
7.6.2.1	<a href="#">lsredis_preset_list_t</a>	225
7.6.3	<a href="#">Function Documentation</a>	225
7.6.3.1	<a href="#">_lsredis_get_obj</a>	225

7.6.3.2	<code>_lsredis_set_value</code>	226
7.6.3.3	<code>lsredis_addRead</code>	227
7.6.3.4	<code>lsredis_addWrite</code>	227
7.6.3.5	<code>lsredis_cleanup</code>	228
7.6.3.6	<code>lsredis_cmpnstr</code>	228
7.6.3.7	<code>lsredis_cmpstr</code>	228
7.6.3.8	<code>lsredis_config</code>	228
7.6.3.9	<code>lsredis_configCB</code>	229
7.6.3.10	<code>lsredis_debugCB</code>	230
7.6.3.11	<code>lsredis_delRead</code>	231
7.6.3.12	<code>lsredis_delWrite</code>	231
7.6.3.13	<code>lsredis_fd_service</code>	231
7.6.3.14	<code>lsredis_find_preset</code>	232
7.6.3.15	<code>lsredis_find_preset_index_by_position</code>	232
7.6.3.16	<code>lsredis_get_obj</code>	232
7.6.3.17	<code>lsredis_get_or_set_d</code>	233
7.6.3.18	<code>lsredis_get_or_set_l</code>	233
7.6.3.19	<code>lsredis_get_string_array</code>	234
7.6.3.20	<code>lsredis_getb</code>	234
7.6.3.21	<code>lsredis_getc</code>	234
7.6.3.22	<code>lsredis_getd</code>	235
7.6.3.23	<code>lsredis_getl</code>	235
7.6.3.24	<code>lsredis_getstr</code>	235
7.6.3.25	<code>lsredis_hgetCB</code>	235
7.6.3.26	<code>lsredis_init</code>	236
7.6.3.27	<code>lsredis_keysCB</code>	237
7.6.3.28	<code>lsredis_load_presets</code>	237
7.6.3.29	<code>lsredis_maybe_add_key</code>	238
7.6.3.30	<code>lsredis_reexec</code>	239
7.6.3.31	<code>lsredis_run</code>	239
7.6.3.32	<code>lsredis_set_preset</code>	239
7.6.3.33	<code>lsredis_set_value</code>	240
7.6.3.34	<code>lsredis_setstr</code>	240
7.6.3.35	<code>lsredis_sig_service</code>	241
7.6.3.36	<code>lsredis_subCB</code>	241
7.6.3.37	<code>lsredis_worker</code>	243
7.6.3.38	<code>redisDisconnectCB</code>	244
7.6.4	<code>Variable Documentation</code>	244
7.6.4.1	<code>lsredis_cond</code>	244
7.6.4.2	<code>lsredis_config_cond</code>	244

7.6.4.3	<a href="#">lsredis_config_mutex</a>	244
7.6.4.4	<a href="#">lsredis_head</a>	244
7.6.4.5	<a href="#">lsredis_htab</a>	244
7.6.4.6	<a href="#">lsredis_key_select_regex</a>	244
7.6.4.7	<a href="#">lsredis_mutex</a>	244
7.6.4.8	<a href="#">lsredis_objs</a>	244
7.6.4.9	<a href="#">lsredis_preset_ht</a>	245
7.6.4.10	<a href="#">lsredis_preset_list</a>	245
7.6.4.11	<a href="#">lsredis_preset_list_mutex</a>	245
7.6.4.12	<a href="#">lsredis_preset_max_n</a>	245
7.6.4.13	<a href="#">lsredis_preset_n</a>	245
7.6.4.14	<a href="#">lsredis_publisher</a>	245
7.6.4.15	<a href="#">lsredis_running</a>	245
7.6.4.16	<a href="#">lsredis_thread</a>	245
7.6.4.17	<a href="#">mutex_initializer</a>	245
7.6.4.18	<a href="#">roac</a>	245
7.6.4.19	<a href="#">rofd</a>	245
7.6.4.20	<a href="#">subac</a>	245
7.6.4.21	<a href="#">subfd</a>	246
7.6.4.22	<a href="#">wrac</a>	246
7.6.4.23	<a href="#">wrfd</a>	246
7.7	<a href="#">lctest.c File Reference</a>	246
7.7.1	<a href="#">Function Documentation</a>	246
7.7.1.1	<a href="#">lctest_lspmac_est_move_time</a>	246
7.7.1.2	<a href="#">lctest_main</a>	247
7.8	<a href="#">lctimer.c File Reference</a>	248
7.8.1	<a href="#">Detailed Description</a>	249
7.8.2	<a href="#">Macro Definition Documentation</a>	249
7.8.2.1	<a href="#">LSTIMER_LIST_LENGTH</a>	249
7.8.2.2	<a href="#">LSTIMER_RESOLUTION_NSECS</a>	249
7.8.3	<a href="#">Typedef Documentation</a>	249
7.8.3.1	<a href="#">lctimer_list_t</a>	249
7.8.4	<a href="#">Function Documentation</a>	250
7.8.4.1	<a href="#">handler</a>	250
7.8.4.2	<a href="#">lctimer_init</a>	250
7.8.4.3	<a href="#">lctimer_run</a>	250
7.8.4.4	<a href="#">lctimer_set_timer</a>	250
7.8.4.5	<a href="#">lctimer_unset_timer</a>	251
7.8.4.6	<a href="#">lctimer_worker</a>	251
7.8.4.7	<a href="#">service_timers</a>	252

7.8.5	Variable Documentation . . . . .	253
7.8.5.1	Istimer_active_timers . . . . .	253
7.8.5.2	Istimer_cond . . . . .	254
7.8.5.3	Istimer_list . . . . .	254
7.8.5.4	Istimer_mutex . . . . .	254
7.8.5.5	Istimer_thread . . . . .	254
7.8.5.6	Istimer_timerid . . . . .	254
7.8.5.7	new_timer . . . . .	254
7.9	md2cmds.c File Reference . . . . .	254
7.9.1	Detailed Description . . . . .	257
7.9.2	Typedef Documentation . . . . .	257
7.9.2.1	md2cmds_cmd_kv_t . . . . .	257
7.9.3	Function Documentation . . . . .	257
7.9.3.1	md2cmds_abort . . . . .	257
7.9.3.2	md2cmds_action_queue . . . . .	257
7.9.3.3	md2cmds_action_wait . . . . .	258
7.9.3.4	md2cmds_collect . . . . .	258
7.9.3.5	md2cmds_coordsys_1_stopped_cb . . . . .	263
7.9.3.6	md2cmds_coordsys_2_stopped_cb . . . . .	263
7.9.3.7	md2cmds_coordsys_3_stopped_cb . . . . .	263
7.9.3.8	md2cmds_coordsys_4_stopped_cb . . . . .	263
7.9.3.9	md2cmds_coordsys_5_stopped_cb . . . . .	263
7.9.3.10	md2cmds_coordsys_7_stopped_cb . . . . .	263
7.9.3.11	md2cmds_home_prep . . . . .	264
7.9.3.12	md2cmds_home_wait . . . . .	264
7.9.3.13	md2cmds_init . . . . .	264
7.9.3.14	md2cmds_is_moving . . . . .	265
7.9.3.15	md2cmds_kappaphi_move . . . . .	266
7.9.3.16	md2cmds_maybe_done_homing_cb . . . . .	266
7.9.3.17	md2cmds_maybe_done_moving_cb . . . . .	266
7.9.3.18	md2cmds_maybe_rotate_done_cb . . . . .	267
7.9.3.19	md2cmds_move_prep . . . . .	267
7.9.3.20	md2cmds_move_wait . . . . .	267
7.9.3.21	md2cmds_moveAbs . . . . .	268
7.9.3.22	md2cmds_moveRel . . . . .	269
7.9.3.23	md2cmds_mvcenter_move . . . . .	270
7.9.3.24	md2cmds_organs_move_presets . . . . .	271
7.9.3.25	md2cmds_phase_beamLocation . . . . .	272
7.9.3.26	md2cmds_phase_center . . . . .	272
7.9.3.27	md2cmds_phase_change . . . . .	273

7.9.3.28	md2cmds_phase_dataCollection . . . . .	274
7.9.3.29	md2cmds_phase_manualMount . . . . .	275
7.9.3.30	md2cmds_phase_robotMount . . . . .	275
7.9.3.31	md2cmds_phase_safe . . . . .	276
7.9.3.32	md2cmds_prep_axis . . . . .	277
7.9.3.33	md2cmds_push_queue . . . . .	277
7.9.3.34	md2cmds_rotate . . . . .	277
7.9.3.35	md2cmds_rotate_cb . . . . .	279
7.9.3.36	md2cmds_run . . . . .	280
7.9.3.37	md2cmds_run_cmd . . . . .	280
7.9.3.38	md2cmds_set . . . . .	281
7.9.3.39	md2cmds_set_scale_cb . . . . .	282
7.9.3.40	md2cmds_settransferpoint . . . . .	282
7.9.3.41	md2cmds_test . . . . .	283
7.9.3.42	md2cmds_time_capz_cb . . . . .	283
7.9.3.43	md2cmds_transfer . . . . .	284
7.9.3.44	md2cmds_worker . . . . .	286
7.9.4	Variable Documentation . . . . .	287
7.9.4.1	md2cmds_capz_moving_time . . . . .	287
7.9.4.2	md2cmds_cmd . . . . .	287
7.9.4.3	md2cmds_cmd_kvs . . . . .	287
7.9.4.4	md2cmds_cmd_pg_kvs . . . . .	288
7.9.4.5	md2cmds_cmd_regex . . . . .	288
7.9.4.6	md2cmds_cond . . . . .	288
7.9.4.7	md2cmds_hmap . . . . .	288
7.9.4.8	md2cmds_homing_cond . . . . .	288
7.9.4.9	md2cmds_homing_count . . . . .	288
7.9.4.10	md2cmds_homing_mutex . . . . .	288
7.9.4.11	md2cmds_md_status_code . . . . .	288
7.9.4.12	md2cmds_moving_cond . . . . .	288
7.9.4.13	md2cmds_moving_count . . . . .	289
7.9.4.14	md2cmds_moving_mutex . . . . .	289
7.9.4.15	md2cmds_moving_queue_wait . . . . .	289
7.9.4.16	md2cmds_mutex . . . . .	289
7.9.4.17	md2cmds_thread . . . . .	289
7.9.4.18	rotating . . . . .	289
7.10	mk_pgpmac_redis.py File Reference . . . . .	289
7.11	pgpmac.c File Reference . . . . .	290
7.11.1	Detailed Description . . . . .	291
7.11.2	Macro Definition Documentation . . . . .	291

---

7.11.2.1	PGPMAC_COMMAND_LINE_LENGTH . . . . .	291
7.11.2.2	PGPMAC_N_COMMAND_LINES . . . . .	291
7.11.3	Function Documentation . . . . .	292
7.11.3.1	main . . . . .	292
7.11.3.2	pgpmac_printf . . . . .	294
7.11.3.3	pgpmac_quit_cb . . . . .	294
7.11.3.4	pgpmac_request_stay_of_execution . . . . .	295
7.11.3.5	stdinService . . . . .	295
7.11.4	Variable Documentation . . . . .	297
7.11.4.1	doomsday_count . . . . .	297
7.11.4.2	doomsday_mutex . . . . .	297
7.11.4.3	ncurses_mutex . . . . .	297
7.11.4.4	pgpmac_use_autoscint . . . . .	298
7.11.4.5	pgpmac_use_pg . . . . .	298
7.11.4.6	running . . . . .	298
7.11.4.7	stdinfda . . . . .	298
7.11.4.8	term_input . . . . .	298
7.11.4.9	term_output . . . . .	298
7.11.4.10	term_status . . . . .	298
7.11.4.11	term_status2 . . . . .	298
7.12	pgpmac.h File Reference . . . . .	298
7.12.1	Detailed Description . . . . .	307
7.12.2	Macro Definition Documentation . . . . .	307
7.12.2.1	_GNU_SOURCE . . . . .	307
7.12.2.2	LS_DISPLAY_WINDOW_HEIGHT . . . . .	307
7.12.2.3	LS_DISPLAY_WINDOW_WIDTH . . . . .	307
7.12.2.4	LS_PG_QUERY_STRING_LENGTH . . . . .	307
7.12.2.5	LSEVENTS_EVENT_LENGTH . . . . .	307
7.12.2.6	LSPMAC_MAGIC_NUMBER . . . . .	308
7.12.2.7	MD2CMDS_CMD_LENGTH . . . . .	308
7.12.3	Typedef Documentation . . . . .	308
7.12.3.1	lspg_demandairrights_t . . . . .	308
7.12.3.2	lspg_getcenter_t . . . . .	308
7.12.3.3	lspg_getcurrentsampleid_t . . . . .	308
7.12.3.4	lspg_nexsample_t . . . . .	308
7.12.3.5	lspg_nextshot_t . . . . .	308
7.12.3.6	lspg_query_queue_t . . . . .	308
7.12.3.7	lspg_starttransfer_t . . . . .	308
7.12.3.8	lspg_waitcryo_t . . . . .	308
7.12.3.9	lspmac_bi_t . . . . .	308

7.12.3.10 <code>lspmac_motor_t</code>	308
7.12.3.11 <code>lsredis_obj_t</code>	309
7.12.3.12 <code>pmac_cmd_queue_t</code>	309
7.12.3.13 <code>pmac_cmd_t</code>	309
7.12.4 Function Documentation	309
7.12.4.1 <code>_lsredis_get_obj</code>	309
7.12.4.2 <code>lsevents_add_listener</code>	310
7.12.4.3 <code>lsevents_init</code>	311
7.12.4.4 <code>lsevents_preregister_event</code>	311
7.12.4.5 <code>lsevents_remove_listener</code>	312
7.12.4.6 <code>lsevents_run</code>	312
7.12.4.7 <code>lsevents_send_event</code>	313
7.12.4.8 <code>lslogging_init</code>	313
7.12.4.9 <code>lslogging_log_message</code>	313
7.12.4.10 <code>lslogging_run</code>	314
7.12.4.11 <code>lspg_array2ptrs</code>	314
7.12.4.12 <code>lspg_demandairrights_all</code>	316
7.12.4.13 <code>lspg_getcenter_call</code>	316
7.12.4.14 <code>lspg_getcenter_done</code>	316
7.12.4.15 <code>lspg_getcenter_wait</code>	316
7.12.4.16 <code>lspg_getcurrentsampleid_wait_for_id</code>	317
7.12.4.17 <code>lspg_init</code>	317
7.12.4.18 <code>lspg_nexsample_all</code>	317
7.12.4.19 <code>lspg_nextshot_call</code>	317
7.12.4.20 <code>lspg_nextshot_done</code>	318
7.12.4.21 <code>lspg_nextshot_wait</code>	318
7.12.4.22 <code>lspg_query_push</code>	318
7.12.4.23 <code>lspg_run</code>	319
7.12.4.24 <code>lspg_seq_run_prep_all</code>	319
7.12.4.25 <code>lspg_starttransfer_call</code>	320
7.12.4.26 <code>lspg_starttransfer_done</code>	320
7.12.4.27 <code>lspg_starttransfer_wait</code>	320
7.12.4.28 <code>lspg_waitcryo_all</code>	320
7.12.4.29 <code>lspg_waitcryo_cb</code>	320
7.12.4.30 <code>lspg_zoom_lut_call</code>	321
7.12.4.31 <code>lspmac_abort</code>	321
7.12.4.32 <code>lspmac_est_move_time</code>	321
7.12.4.33 <code>lspmac_est_move_time_wait</code>	326
7.12.4.34 <code>lspmac_find_motor_by_name</code>	327
7.12.4.35 <code>lspmac_getBIPosition</code>	327

7.12.4.36 <code>lspmac_getPosition</code>	327
7.12.4.37 <code>lspmac_home1_queue</code>	328
7.12.4.38 <code>lspmac_home2_queue</code>	329
7.12.4.39 <code>lspmac_init</code>	330
7.12.4.40 <code>lspmac_jogabs_queue</code>	333
7.12.4.41 <code>lspmac_move_or_jog_abs_queue</code>	333
7.12.4.42 <code>lspmac_move_or_jog_preset_queue</code>	336
7.12.4.43 <code>lspmac_move_or_jog_queue</code>	337
7.12.4.44 <code>lspmac_move_preset_queue</code>	337
7.12.4.45 <code>lspmac_moveabs_queue</code>	337
7.12.4.46 <code>lspmac_moveabs_wait</code>	338
7.12.4.47 <code>lspmac_run</code>	339
7.12.4.48 <code>lspmac_set_motion_flags</code>	340
7.12.4.49 <code>lspmac_SockSendControlCharPrint</code>	342
7.12.4.50 <code>lspmac_SockSendDPControlChar</code>	342
7.12.4.51 <code>lspmac_SockSendDPLine</code>	342
7.12.4.52 <code>lspmac_SockSendline</code>	342
7.12.4.53 <code>lspmac_video_rotate</code>	343
7.12.4.54 <code>lsredis_cmpnstr</code>	343
7.12.4.55 <code>lsredis_cmpstr</code>	344
7.12.4.56 <code>lsredis_config</code>	344
7.12.4.57 <code>lsredis_find_preset</code>	344
7.12.4.58 <code>lsredis_find_preset_index_by_position</code>	345
7.12.4.59 <code>lsredis_get_obj</code>	345
7.12.4.60 <code>lsredis_get_string_array</code>	346
7.12.4.61 <code>lsredis_getb</code>	346
7.12.4.62 <code>lsredis_getc</code>	346
7.12.4.63 <code>lsredis_getd</code>	346
7.12.4.64 <code>lsredis_getl</code>	347
7.12.4.65 <code>lsredis_getstr</code>	347
7.12.4.66 <code>lsredis_init</code>	347
7.12.4.67 <code>lsredis_load_presets</code>	348
7.12.4.68 <code>lsredis_reexec</code>	349
7.12.4.69 <code>lsredis_run</code>	350
7.12.4.70 <code>lsredis_set_preset</code>	350
7.12.4.71 <code>lsredis_setstr</code>	351
7.12.4.72 <code>lptest_main</code>	351
7.12.4.73 <code>lstimer_init</code>	352
7.12.4.74 <code>lstimer_run</code>	352
7.12.4.75 <code>lstimer_set_timer</code>	352

7.12.4.76 lsTimer_unset_timer . . . . .	353
7.12.4.77 lsupdate_init . . . . .	353
7.12.4.78 lsupdate_run . . . . .	353
7.12.4.79 md2cmds_init . . . . .	353
7.12.4.80 md2cmds_push_queue . . . . .	354
7.12.4.81 md2cmds_run . . . . .	355
7.12.4.82 pgpmac_printf . . . . .	355
7.12.4.83 pgpmac_request_stay_of_execution . . . . .	355
7.12.4.84 PmacSockSendline . . . . .	356
7.12.5 Variable Documentation . . . . .	356
7.12.5.1 alignx . . . . .	356
7.12.5.2 aligny . . . . .	356
7.12.5.3 alignz . . . . .	356
7.12.5.4 anal . . . . .	356
7.12.5.5 apery . . . . .	356
7.12.5.6 aperz . . . . .	356
7.12.5.7 arm_parked . . . . .	356
7.12.5.8 blight . . . . .	356
7.12.5.9 blight_down . . . . .	356
7.12.5.10 blight_f . . . . .	357
7.12.5.11 blight_ud . . . . .	357
7.12.5.12 blight_up . . . . .	357
7.12.5.13 capy . . . . .	357
7.12.5.14 capz . . . . .	357
7.12.5.15 cenx . . . . .	357
7.12.5.16 ceny . . . . .	357
7.12.5.17 cryo . . . . .	357
7.12.5.18 cryo_back . . . . .	357
7.12.5.19 cryo_switch . . . . .	358
7.12.5.20 dryer . . . . .	358
7.12.5.21 etel_init_ok . . . . .	358
7.12.5.22 etel_on . . . . .	358
7.12.5.23 etel_ready . . . . .	358
7.12.5.24 flight . . . . .	358
7.12.5.25 flight_f . . . . .	358
7.12.5.26 flight_oo . . . . .	358
7.12.5.27 fluo . . . . .	358
7.12.5.28 fluor_back . . . . .	359
7.12.5.29 fscint . . . . .	359
7.12.5.30 fshut . . . . .	359

7.12.5.31 hp_air . . . . .	359
7.12.5.32 kappa . . . . .	359
7.12.5.33 lp_air . . . . .	359
7.12.5.34 lsgp_demandairrights . . . . .	359
7.12.5.35 lsgp_getcenter . . . . .	359
7.12.5.36 lsgp_getcurrentsampleid . . . . .	359
7.12.5.37 lsgp_nexsample . . . . .	360
7.12.5.38 lsgp_nextshot . . . . .	360
7.12.5.39 lsgp_starttransfer . . . . .	360
7.12.5.40 lsgp_waitcryo . . . . .	360
7.12.5.41 lspmac_motors . . . . .	360
7.12.5.42 lspmac_moving_cond . . . . .	360
7.12.5.43 lspmac_moving_flags . . . . .	360
7.12.5.44 lspmac_moving_mutex . . . . .	360
7.12.5.45 lspmac_nmotors . . . . .	360
7.12.5.46 lspmac_shutter_cond . . . . .	361
7.12.5.47 lspmac_shutter_has_opened . . . . .	361
7.12.5.48 lspmac_shutter_mutex . . . . .	361
7.12.5.49 lspmac_shutter_state . . . . .	361
7.12.5.50 lsredis_cond . . . . .	361
7.12.5.51 lsredis_mutex . . . . .	361
7.12.5.52 lsredis_running . . . . .	361
7.12.5.53 md2_status_mutex . . . . .	361
7.12.5.54 md2cmds_cmd . . . . .	361
7.12.5.55 md2cmds_cond . . . . .	361
7.12.5.56 md2cmds_md_status_code . . . . .	362
7.12.5.57 md2cmds_mutex . . . . .	362
7.12.5.58 md2cmds_pg_cond . . . . .	362
7.12.5.59 md2cmds_pg_mutex . . . . .	362
7.12.5.60 minikappa_ok . . . . .	362
7.12.5.61 ncurses_mutex . . . . .	362
7.12.5.62 omega . . . . .	362
7.12.5.63 omega_zero_time . . . . .	362
7.12.5.64 pgpmac_use_autoscint . . . . .	362
7.12.5.65 pgpmac_use_pg . . . . .	362
7.12.5.66 phi . . . . .	362
7.12.5.67 pmac_queue_cond . . . . .	363
7.12.5.68 pmac_queue_mutex . . . . .	363
7.12.5.69 sample_detected . . . . .	363
7.12.5.70 scint . . . . .	363

---

7.12.5.71 shutter_open . . . . .	363
7.12.5.72 smart_mag_err . . . . .	363
7.12.5.73 smart_mag_off . . . . .	363
7.12.5.74 smart_mag_on . . . . .	363
7.12.5.75 smart_mag_oo . . . . .	363
7.12.5.76 term_input . . . . .	364
7.12.5.77 term_output . . . . .	364
7.12.5.78 term_status . . . . .	364
7.12.5.79 term_status2 . . . . .	364
7.12.5.80 zoom . . . . .	364

<b>Index</b>	<b>364</b>
--------------	------------



# Chapter 1

## The LS-CAT pgpmac Project

### pgpmac.c

Some pmac defines, typedefs, functions suggested by Delta Tau Accessory 54E User Manual, October 23, 2003 (C) 2003 by Delta Tau Data Systems, Inc. All rights reserved.

Original work Copyright (C) 2012 by Keith Brister, Northwestern University, All rights reserved.

This project implements the MD2 communications required for operation at LS-CAT and is intended to replace Windows XP based .NET code provided by MAATEL.

The need to do this is driven by a desire to make the system as efficient and fast as possible by combining various operations. A proof-of-principle version of this code saw frame rates of 23/minute as opposed to the nominal 18/minute we normally quote for 1 second exposures.

Additionally, as we rapidly approach EOL for Windows XP an alternative is urgently needed.

### Structure

The project is roughly broken down as follows:

lsevents.c	Simple event queue
lsredis.c	Receive key value pair updates from redis databases
lslogging.c	A logging utility to simplify debugging
lspg.c	Handles communications with the controlling postgresql database
lsupdate.c	Periodically update the px.kvs table with new positions.
md2cmds.c	Provides the equivalent (mostly) of the LS-CAT BLUMax code.
pgpmac.c	Main: parses command line and starts up the various threads
pgpmac.h	All includes and defines. The only file included by the .c files in this
pmac_md2_ls-cat.pmc	Code for the PMAC: compile and install with pmac executable program.
pmac_md2.sql	Tables and procedures for the postgresql side of the project.

### Notes:

- The postgresql and the pmac communications interfaces are asynchronous and rely heavily on the unix "poll" routine.
- The project is multithreaded and based on "pthreads".
- Most threads maintain a queue of commands to simplify communications with each other.
- Note that a MAATEL supported interface for a more recent version of Windows may be available, however, a bit of effort will be required to implement it at LS-CAT as the BLUMax code will likely require some revisions. This is still an option should the present project become intractable.
- An important constraint has been to run the MD2 either from the windows .NET environment or from the pgpmac environment. A consequence is that the pmac "pmc" file has been augmented to include new capabilities without destroying the code that the .NET interface requires.
- Epics support could come by adapting the "e.c" code to work here directly or could come by making use of the existing kv pair mechanism already in place or, as is most likely, a combination of the two.
- Ncurses support could include input lines for SQL queries and direct commands for supporting homing etc. Perhaps the F keys could change modes or use of special mode changing text commands. Output is not asynchronous. Although this is unlikely to cause a problem I'd hate to have the program hang because terminal output is hung up.
- PG queries come back as text instead of binary. We could reduce the numeric errors by using binary and things would run a tad faster, though it is unlikely anyone would notice or care about the speed.

## MD2 Motors and Coordinate Systems

CS            Motor

```

1            1        X = Omega

2            17      X = Center X
18          Y = Center Y

3            2        X = Alignment X
3            Y = Alignment Y
4            Z = Alignment Z

--          5        Analyzer

4            6        X = Zoom

5            7        Y = Aperture Y
8            Z = Aperture Z
9            U = Capillary Y
10          V = Capillary Z
11          W = Scintillator Z

6                    (None)

7            19      X = Kappa
20          Y = Phi

```

## MD2 Motion Programs

---

```

before calling, set
M4XX = 1: flag to indicate we are running program XX
P variables as arguments

Program          Description
1               home omega
2               home alignment table X
3               home alignment table Y
4               home alignment table Z
6               home camera zoom
7               home aperture Y
8               home aperture Z
9               home capillary Y
10              home capillary Z
11              home scintillator Z
17              home center X
18              home center Y
19              home kappa
20              home phi (Home position is not defined for phi ...)
25              kappa stress test

26              Combined Incremental move of X and Y in selected coordinate system
                  (Does not reset M426)
                  P170 = X increment
                  P171 = Y increment

31              scan omega
                  P170 = Start
                  P171 = End
                  P173 = Velocity (float)
                  P174 = Sample Rate (I5049)
                  P175 = Acceleration time
                  P176 = Gathering source
                  P177 = Number of passes
                  P178 = Shutter rising distance (units of omega motion)
                  P179 = Shutter falling distance (units of omega motion)
                  P180 = Exposure Time

34              Organ Scan
                  P169 = Motor Number
                  P170 = Start Position
                  P171 = End Position
                  P172 = Step Size
                  P173 = Motor Speed

35              Organ Homing

37              Organ Move (microdiff_hard.ini says we don't use this anymore)
                  P169 = Capillary Z
                  P170 = Scintillator Z
                  P171 = Aperture Z

50              Combined Incremental move of X and Y
                  P170 = X increment
                  P171 = Y increment

52              X oscillation (while M320 == 1)
                  (Does not reset M452)

53              Center X and Y Synchronized homing

```

---

54                   Combined X, Y, Z absolute move  
P170        = X  
P171        = Y  
P172        = Z

131                  LS-CAT Modified Omega Scan  
P170        = Shutter open position, in counts  
P171        = Delta omega, in counts  
P173        = Omega velocity (counts/msec)  
P175        = Acceleration Time (msec)  
P177        = Number of passes  
P178        = Shutter Rising Distance  
P179        = Shutter Falling Distance  
P180        = Exposure TIme (msec)

140                  LS-CAT Move X Absolute  
Q10         = X Value (cts)

141                  LS-CAT Move Y Absolute  
Q11         = Y Value (cts)

142                  LS-CAT Move Z Absolute  
Q12         = Z Value (cts)

150                  LS-CAT Move X, Y Absolute  
Q20         = X Value  
Q21         = Y Value

160                  LS-CAT Move X, Y, Z Absolute  
Q30         = X Value  
Q31         = Y Value  
Q32         = Z Value

# Chapter 2

## Namespace Index

### 2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

<a href="#">iniParser</a> . . . . .	11
<a href="#">mk_pgpmac_redis</a> . . . . .	11



## Chapter 3

# Data Structure Index

### 3.1 Data Structures

Here are the data structures with brief descriptions:

<a href="#">iniParser.iniParser</a>	This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version . . . . .	17
<a href="#">lsevents_callbacks_struct</a>	Lsevents linked list of callbacks for each event . . . . .	19
<a href="#">lsevents_event_names_struct</a>	Linked list of all the event names used to regenerate the hash table . . . . .	20
<a href="#">lsevents_listener_struct</a>	Linked list of event listeners . . . . .	21
<a href="#">lsevents_queue_struct</a>	Storage definition for the events . . . . .	22
<a href="#">lslogging_queue_struct</a>	Our log object: time and message . . . . .	22
<a href="#">lspg_demandairrights_struct</a>	. . . . .	23
<a href="#">lspg_getcenter_struct</a>	Storage for getcenter query Used for the md2 ROTATE command that generates the centering movies . . . . .	24
<a href="#">lspg_getcurrentsampleid_struct</a>	. . . . .	26
<a href="#">lspg_lock_detector_struct</a>	Lock detector object Implements detector lock for exposure control . . . . .	27
<a href="#">lspg_lock_diffractometer_struct</a>	Object used to implement locking the diffractometer Critical to exposure timing . . . . .	28
<a href="#">lspg_nextsample_struct</a>	Returns the next sample number Just a 32 bit int (Hal, take that, nextshot!) . . . . .	29
<a href="#">lspg_nextshot_struct</a>	Storage definition for nextshot query . . . . .	30
<a href="#">lspg_seq_run_prep_struct</a>	Data collection running object . . . . .	41
<a href="#">lspg_starttransfer_struct</a>	Returns 1 if transfer can continue 0 to abort . . . . .	42
<a href="#">lspg_wait_for_detector_struct</a>	Object that implements detector / spindle timing We use database locks for exposure control and this implements the md2 portion of this handshake . . . . .	43
<a href="#">lspg_waitcryo_struct</a>	. . . . .	43
<a href="#">lspgQueryQueueStruct</a>	Store each query along with its callback function . . . . .	44
<a href="#">lspmac_ascii_buffers_struct</a>	. . . . .	45

<a href="#">lspmac.bi_struct</a>	
Storage for binary inputs	46
<a href="#">lspmac.cmd_queue_struct</a>	
PMAC command queue item	48
<a href="#">lspmac.combined_move_struct</a>	
	49
<a href="#">lspmac.dpascii_queue_struct</a>	
	49
<a href="#">lspmac.motor_struct</a>	
Motor information	50
<a href="#">lsredis.obj_struct</a>	
Redis Object Basic object whose value is synchronized with our redis db	58
<a href="#">lsredis.preset_list_struct</a>	
	61
<a href="#">lstopic.list_struct</a>	
Everything we need to know about a timer	62
<a href="#">md2cmds.cmd_kv_struct</a>	
	64
<a href="#">md2StatusStruct</a>	
The block of memory retrieved in a status request	64
<a href="#">tagEthernetCmd</a>	
PMAC ethernet packet definition	72

# Chapter 4

## File Index

### 4.1 File List

Here is a list of all files with brief descriptions:

iniParser.py	75
lsevents.c	
Event subsystem for inter-pgpmac communication	75
lslogging.c	
Logs messages to a file	84
lspg.c	
Postgresql support for the LS-CAT pgpmac project	89
lspmac.c	
Routines concerned with communication with PMAC	133
lsredis.c	
Support redis hash synchronization	222
ltest.c	
lstimer.c	
Support for delayed and periodic events	248
md2cmds.c	
Implements commands to run the md2 diffractometer attached to a PMAC controled by post-gresql	254
mk_pgpmac_redis.py	
pgpmac.c	
Main for the pgpmac project	290
pgpmac.h	
Headers for the entire pgpmac project	298



# Chapter 5

## Namespace Documentation

### 5.1 iniParser Namespace Reference

#### Data Structures

- class `iniParser`

*This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.*

#### Variables

- tuple `ip iniParser( "21-ID-D/microdiff_pref.ini")`

#### 5.1.1 Variable Documentation

##### 5.1.1.1 tuple `iniParser.ip iniParser( "21-ID-D/microdiff_pref.ini")`

Definition at line 104 of file `iniParser.py`.

### 5.2 mk\_pgpmac\_redis Namespace Reference

#### Functions

- def `mk_home`
- def `mk_active_init`
- def `mk_inactive_init`
- def `active_simulation`
- def `asis`

#### Variables

- list `head` `sys.argv[1]`
- list `pref_ini` `sys.argv[2]`
- list `hard_ini` `sys.argv[3]`
- dictionary `configs`
- dictionary `plcc2_dict`

- dictionary `motor_dict`
- dictionary `hard_ini_fields`
- list `motor_field_lists`
- list `bi_list` ["CryoSwitch"]
- dictionary `motor_presets`
- list `zoom_settings`
- tuple `hi iniParser.iniParser( hard_ini )`
- list `v motor_dict[m]`
- string `f "HSETNX"`
- list `xlate hard_ini_fields[k]`
- tuple `pi iniParser.iniParser( pref_ini )`
- int `i 0`
- tuple `ppos pi.get( section, option )`
- string `fnc "HSETNX"`
- tuple `b pi.get( section, "LightIntensity" )`
- tuple `p pi.get( section, "MotorPosition" )`
- tuple `x pi.get( section, "ScaleX" )`
- tuple `y pi.get( section, "ScaleY" )`
- tuple `plcc2_file open( "%s-plcc2.pmc" % (head), "w" )`
- tuple `motor_num int( motor_dict[m]["motor_num"] )`

## 5.2.1 Function Documentation

### 5.2.1.1 def mk\_pgpmac\_redis.active\_simulation ( sim )

Definition at line 424 of file `mk_pgpmac_redis.py`.

```
424
425 def active_simulation( sim ):
426     if str(sim) != "0":
427         rtn = "0"
428     else:
429         rtn = "1"
430     return rtn
```

### 5.2.1.2 def mk\_pgpmac\_redis.asis ( arg )

Definition at line 431 of file `mk_pgpmac_redis.py`.

```
431
432 def asis( arg ):
433     return arg
```

### 5.2.1.3 def mk\_pgpmac\_redis.mk\_active\_init ( d )

Definition at line 402 of file `mk_pgpmac_redis.py`.

```
402
403 def mk_active_init( d ):
404     if not d.has_key("motor_num") or not d.has_key("coord_num") or not
405     d.has_key( "axis" ):
406         return ""
407     motor_num = int(d["motor_num"])
408     coord_num = int(d["coord_num"])
409     axis      = str(d["axis"])
410     mask      = 1 << (motor_num - 1)
411     if motor_num < 1 or motor_num > 32:
412         return ""
413     return '{M%d=1,&%d#%d->%s,"M700=(M700 | $%0x) ^ $%0x"}' % (motor_num + 30,
414     coord_num, motor_num, axis, mask, mask)
```

## 5.2.1.4 def mk\_pgpmac\_redis.mk\_home( mname, d )

Definition at line 387 of file mk\_pgpmac\_redis.py.

```
387
388 def mk_home( mname, d ):
389     if not d.has_key("motor_num") or not d.has_key("coord_num"):
390         return ""
391     motor_num = int(d["motor_num"])
392     coord_num = int(d["coord_num"])
393     if motor_num < 1 or motor_num > 32:
394         return ""
395     if mname == "kappa":
396         prog_num = 119
397     else:
398         prog_num = motor_num
399
400
401     return '#%d$,M%d=1,&%dE,%#d&%dB%dR)' % (motor_num, motor_num+400,
coord_num, motor_num, coord_num, prog_num)
```

## 5.2.1.5 def mk\_pgpmac\_redis.mk\_inactive\_init( d )

Definition at line 413 of file mk\_pgpmac\_redis.py.

```
413
414 def mk_inactive_init( d ):
415     if not d.has_key("motor_num") or not d.has_key("coord_num") or not
d.has_key( "axis" ):
416         return ""
417     motor_num = int(d["motor_num"])
418     coord_num = int(d["coord_num"])
419     axis      = str(d["axis"])
420     mask      = 1 << (motor_num - 1)
421     if motor_num < 1 or motor_num > 32:
422         return ""
423     return '{M%d=0,&%d#%d->0,"M700=M700 | $%0x"}' % (motor_num + 30, coord_num,
motor_num, mask)
```

## 5.2.2 Variable Documentation

## 5.2.2.1 tuple mk\_pgpmac\_redis.b.pi.get( section, "LightIntensity" )

Definition at line 702 of file mk\_pgpmac\_redis.py.

## 5.2.2.2 list mk\_pgpmac\_redis.bi\_list [ "CryoSwitch" ]

Definition at line 495 of file mk\_pgpmac\_redis.py.

## 5.2.2.3 dictionary mk\_pgpmac\_redis.configs

**Initial value:**

```
1 {
2     "orange-2"          : { "re" : "redis\kvseq\stns\.2\.(.+)", "head" : "
3     stns.2", "pub" : "MD2-21-ID-E", "pg" : "1", "autoscint" : "1"},,
4     "orange-2.ls-cat.org" : { "re" : "redis\kvseq\stns\.2\.(.+)", "head" : "
5     stns.2", "pub" : "MD2-21-ID-E", "pg" : "1", "autoscint" : "1"},,
6     "venison.ls-cat.org" : { "re" : "redis\kvseq\stns\.2\.(.+)", "head" : "
7     stns.2", "pub" : "MD2-21-ID-E", "pg" : "1", "autoscint" : "1"},,
8     "mung-2"            : { "re" : "redis\kvseq\stns\.1\.(.+)", "head" : "
9     stns.1", "pub" : "MD2-21-ID-D", "pg" : "1", "autoscint" : "1"},,
10    "mung-2.ls-cat.org" : { "re" : "redis\kvseq\stns\.1\.(.+)", "head" : "
11    stns.1", "pub" : "MD2-21-ID-D", "pg" : "1", "autoscint" : "1"},,
12    "vidalia.ls-cat.org" : { "re" : "redis\kvseq\stns\.1\.(.+)", "head" : "
13    stns.1", "pub" : "MD2-21-ID-D", "pg" : "1", "autoscint" : "1"},,
```

Definition at line 26 of file mk\_pgpmac\_redis.py.

#### 5.2.2.4 tuple mk\_pgpmac\_redis.f "HSETNX"

Definition at line 633 of file mk\_pgpmac\_redis.py.

#### 5.2.2.5 string mk\_pgpmac\_redis.fnc "HSETNX"

Definition at line 693 of file mk\_pgpmac\_redis.py.

#### 5.2.2.6 mk\_pgpmac\_redis.hard\_ini sys.argv[3]

Definition at line 21 of file mk\_pgpmac\_redis.py.

#### 5.2.2.7 dictionary mk\_pgpmac\_redis.hard\_ini\_fields

##### **Initial value:**

```

1 {
2     "active"          : ["Simulation", active_simulation],
3     "coord_num"       : ["CoordinateSystem", asis],
4     "largeStep"        : ["LargeStep", asis],
5     "maxPosition"      : ["MaxPosition", asis],
6     "minPosition"      : ["MinPosition", asis],
7     "motor_num"        : ["MotorNumber", asis],
8     "neutralPosition" : ["NeutralPosition", asis],
9     "precision"        : ["Precision", asis],
10    "smallStep"        : ["SmallStep", asis],
11    "u2c"              : ["UnitRatio", asis]
12 }
```

Definition at line 434 of file mk\_pgpmac\_redis.py.

#### 5.2.2.8 list mk\_pgpmac\_redis.head sys.argv[1]

Definition at line 13 of file mk\_pgpmac\_redis.py.

#### 5.2.2.9 tuple mk\_pgpmac\_redis.hi iniParser.iniParser( hard\_ini )

Definition at line 589 of file mk\_pgpmac\_redis.py.

#### 5.2.2.10 int mk\_pgpmac\_redis.i 0

Definition at line 657 of file mk\_pgpmac\_redis.py.

#### 5.2.2.11 dictionary mk\_pgpmac\_redis.motor\_dict

Definition at line 257 of file mk\_pgpmac\_redis.py.

#### 5.2.2.12 list mk\_pgpmac\_redis.motor\_field\_lists

Definition at line 456 of file mk\_pgpmac\_redis.py.

5.2.2.13 tuple `mk_pgpmac_redis.motor_num int( motor_dict[m]["motor_num"] )`

Definition at line 741 of file `mk_pgpmac_redis.py`.

5.2.2.14 dictionary `mk_pgpmac_redis.motor_presets`

Definition at line 497 of file `mk_pgpmac_redis.py`.

5.2.2.15 tuple `mk_pgpmac_redis.p pi.get( section, "MotorPosition" )`

Definition at line 709 of file `mk_pgpmac_redis.py`.

5.2.2.16 tuple `mk_pgpmac_redis.pi iniParser.iniParser( pref_ini )`

Definition at line 654 of file `mk_pgpmac_redis.py`.

5.2.2.17 dictionary `mk_pgpmac_redis.plcc2_dict`

#### Initial value:

```

1 {
2     "omega"      : { "status1" : "M5001", "status2" : "M5021", "position" : "
3     M5041" },
4     "align.x"    : { "status1" : "M5002", "status2" : "M5022", "position" : "
5     M5042" },
6     "align.y"    : { "status1" : "M5003", "status2" : "M5023", "position" : "
7     M5043" },
8     "align.z"    : { "status1" : "M5004", "status2" : "M5024", "position" : "
9     M5044" },
10    "lightPolar" : { "status1" : "M5005", "status2" : "M5025", "position" : "
11    M5045" },
12    "cam.zoom"   : { "status1" : "M5006", "status2" : "M5026", "position" : "
13    M5046" },
14    "appy"       : { "status1" : "M5007", "status2" : "M5027", "position" : "
15    M5047" },
16    "appz"       : { "status1" : "M5008", "status2" : "M5028", "position" : "
17    M5048" },
18    "capy"       : { "status1" : "M5009", "status2" : "M5029", "position" : "
19    M5049" },
20    "capz"       : { "status1" : "M5010", "status2" : "M5030", "position" : "
21    M5050" },
22    "scint"      : { "status1" : "M5011", "status2" : "M5031", "position" : "
23    M5051" },
24    "centering.x": { "status1" : "M5012", "status2" : "M5032", "position" : "
25    M5052" },
26    "centering.y": { "status1" : "M5013", "status2" : "M5033", "position" : "
27    M5053" },
28    "kappa"      : { "status1" : "M5014", "status2" : "M5034", "position" : "
29    M5054" },
30    "phi"        : { "status1" : "M5015", "status2" : "M5035", "position" : "
31    M5055" }
32 }
```

Definition at line 35 of file `mk_pgpmac_redis.py`.

5.2.2.18 tuple `mk_pgpmac_redis.plcc2_file open( "%s-plcc2.pmc" % (head), "w" )`

Definition at line 729 of file `mk_pgpmac_redis.py`.

5.2.2.19 tuple `mk_pgpmac_redis.ppos pi.get( section, option )`

Definition at line 665 of file `mk_pgpmac_redis.py`.

### 5.2.2.20 mk\_pgpmac\_redis.pref\_ini sys.argv[2]

Definition at line 16 of file mk\_pgpmac\_redis.py.

### 5.2.2.21 tuple mk\_pgpmac\_redis.v motor\_dict[m]

Definition at line 632 of file mk\_pgpmac\_redis.py.

### 5.2.2.22 tuple mk\_pgpmac\_redis.x pi.get( section, "ScaleX")

Definition at line 716 of file mk\_pgpmac\_redis.py.

### 5.2.2.23 list mk\_pgpmac\_redis.xlate hard\_ini\_fields[k]

Definition at line 637 of file mk\_pgpmac\_redis.py.

### 5.2.2.24 tuple mk\_pgpmac\_redis.y pi.get( section, "ScaleY")

Definition at line 723 of file mk\_pgpmac\_redis.py.

### 5.2.2.25 list mk\_pgpmac\_redis.zoom\_settings

#### Initial value:

```

1 [
2     #lev   front   back   pos      scalex  scaley    section
3     [1,      4.0,    8.0,   34100,  2.7083,  3.3442,  "CoaxCam.Zoom1"],
4     [2,      6.0,    8.1,   31440,  2.2487,  2.2776,  "CoaxCam.Zoom2"],
5     [3,      6.5,    8.2,   27460,  1.7520,  1.7550,  "CoaxCam.Zoom3"],
6     [4,      7.0,    8.3,   23480,  1.3360,  1.3400,  "CoaxCam.Zoom4"],
7     [5,      8.0,    10.0,  19500,  1.0140,  1.0110,  "CoaxCam.Zoom5"],
8     [6,      9.0,    12.0,  15520,  0.7710,  0.7760,  "CoaxCam.Zoom6"],
9     [7,     10.0,   17.0,  11540,  0.5880,  0.5920,  "CoaxCam.Zoom7"],
10    [8,     12.0,   25.0,  7560,   0.4460,  0.4480,  "CoaxCam.Zoom8"],
11    [9,     15.0,   37.0,  3580,   0.3410,  0.3460,  "CoaxCam.Zoom9"],
12    [10,    16.0,   42.0,   0,     0.2700,  0.2690,  "CoaxCam.Zoom10"]
13 ]

```

Definition at line 566 of file mk\_pgpmac\_redis.py.

## Chapter 6

# Data Structure Documentation

### 6.1 iniParser.iniParser Class Reference

This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

#### Public Member Functions

- def `__init__`
- def `read`
- def `sections`
- def `options`
- def `has_section`
- def `has_option`
- def `get`

#### Data Fields

- `f`
- `sd`

#### 6.1.1 Detailed Description

This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program. If not, see <http://www.gnu.org/licenses/>.

We assume the sections and options are case insensitive and that, although nested sections are implied by the format used by the md2, that the nesting has no practical importance.

The current version is for READING the files.

TODO: add writing. We'll need to keep track of the preferred case used in the ini file as well as the existing comments. This is mildly tricky since comments apparently can appear on both option lines and non-option lines so

we'll need to track the line number within each section to preserve all the comments. Strictly speaking this is not necessary as we can just spit stuff out all lower case without comments and, presumably, the md2 should be able to deal with it. However, there is enough of a problem with the lack of documentation that willfully removing seems like a bad idea.

Definition at line 42 of file iniParser.py.

## 6.1.2 Constructor & Destructor Documentation

### 6.1.2.1 def iniParser.iniParser.\_\_init\_\_( self, fn )

Definition at line 44 of file iniParser.py.

```
44
45     def __init__( self, fn):
46         self.f = open( fn, "r")
47         self.sd = {}
48
```

## 6.1.3 Member Function Documentation

### 6.1.3.1 def iniParser.iniParser.get( self, section, option )

Definition at line 99 of file iniParser.py.

```
99
100    def get( self, section, option):
101        return self.sd[section.lower()][option.lower()]
102
```

### 6.1.3.2 def iniParser.iniParser.has\_option( self, section, option )

Definition at line 94 of file iniParser.py.

```
94
95     def has_option( self, section, option):
96         if self.has_section( section):
97             return self.sd[section.lower()].has_key( option.lower())
98         return False
```

### 6.1.3.3 def iniParser.iniParser.has\_section( self, section )

Definition at line 91 of file iniParser.py.

```
91
92     def has_section( self, section):
93         return self.sd.has_key( section.lower())
```

### 6.1.3.4 def iniParser.iniParser.options( self, section )

Definition at line 87 of file iniParser.py.

```
87
88     def options( self, section):
89         return self.sd[section.lower()].keys()
90
```

## 6.1.3.5 def iniParser.iniParser.read( self )

Definition at line 49 of file iniParser.py.

```

49
50     def read( self):
51         self.sd = {}
52         current_section = "default"
53         current_dict   = {}
54         for l in self.f.readlines():
55             sl = l.strip()
56             if len(sl) > 0:
57                 if sl[0] == ";":
58                     continue
59
60                 if sl[0] == "[" and sl.find("]") > 1:
61                     self.sd[current_section] = current_dict
62                     current_dict = {}
63                     current_section = (sl[1:sl.find("[")]).lower()
64
65                 else:
66                     if sl.find(";") > 0:
67                         s = sl[0:sl.find(";")]
68                     else:
69                         s = sl
70
71                     if s.find("=") > 0:
72                         slist = s.split ="")
73                         if len(slist) == 2:
74                             k = (slist[0].strip()).lower()
75                             v = slist[1].strip()
76                             current_dict[k] = v
77
78         self.sd[current_section] = current_dict
79
80     self.f.close()
81
82

```

## 6.1.3.6 def iniParser.iniParser.sections( self )

Definition at line 83 of file iniParser.py.

```

83
84     def sections( self):
85         ks = set(self.sd.keys())
86         return list(ks.difference( ["default"]))

```

## 6.1.4 Field Documentation

## 6.1.4.1 iniParser.iniParser.f

Definition at line 45 of file iniParser.py.

## 6.1.4.2 iniParser.iniParser.sd

Definition at line 46 of file iniParser.py.

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

- [iniParser.py](#)

## 6.2 Ievents\_callbacks\_struct Struct Reference

Ievents linked list of callbacks for each event

## Data Fields

- struct [lsevents\\_callbacks\\_struct](#) \* next
- void(\* [cb](#) )(char \*)

### 6.2.1 Detailed Description

lsevents linked list of callbacks for each event

Definition at line 46 of file lsevents.c.

### 6.2.2 Field Documentation

#### 6.2.2.1 void(\* lsevents\_callbacks\_struct::cb)(char \*)

Definition at line 48 of file lsevents.c.

#### 6.2.2.2 struct lsevents\_callbacks\_struct\* lsevents\_callbacks\_struct::next

Definition at line 47 of file lsevents.c.

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

- [lsevents.c](#)

## 6.3 lsevents\_event\_names\_struct Struct Reference

linked list of all the event names used to regenerate the hash table

## Data Fields

- struct [lsevents\\_event\\_names\\_struct](#) \* next
- char \* [event](#)
- [lsevents\\_callbacks\\_t](#) \* [cbl](#)

### 6.3.1 Detailed Description

linked list of all the event names used to regenerate the hash table

Definition at line 55 of file lsevents.c.

### 6.3.2 Field Documentation

#### 6.3.2.1 [lsevents\\_callbacks\\_t](#)\* lsevents\_event\_names\_struct::[cbl](#)

Definition at line 58 of file lsevents.c.

#### 6.3.2.2 char\* lsevents\_event\_names\_struct::[event](#)

Definition at line 57 of file lsevents.c.

**6.3.2.3 struct lsevents\_event\_names\_struct\* lsevents\_event\_names\_struct::next**

Definition at line 56 of file lsevents.c.

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

- [lsevents.c](#)

## 6.4 lsevents\_listener\_struct Struct Reference

Linked list of event listeners.

### Data Fields

- struct [lsevents\\_listener\\_struct](#) \* **next**  
*Next listener.*
- char \* **raw\_regexp**  
*the original string sent to us*
- regex\_t **re**  
*regular expression representing listened for events*
- void(\* **cb** )(char \*)  
*call back function*

### 6.4.1 Detailed Description

Linked list of event listeners.

Definition at line 35 of file lsevents.c.

### 6.4.2 Field Documentation

#### 6.4.2.1 void(\* lsevents\_listener\_struct::cb)(char \*)

call back function

Definition at line 39 of file lsevents.c.

#### 6.4.2.2 struct lsevents\_listener\_struct\* lsevents\_listener\_struct::next

Next listener.

Definition at line 36 of file lsevents.c.

#### 6.4.2.3 char\* lsevents\_listener\_struct::raw\_regexp

the original string sent to us

Definition at line 37 of file lsevents.c.

#### 6.4.2.4 `regex_t lsevents_listener_struct::re`

regular expression representing listened for events

Definition at line 38 of file lsevents.c.

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

- [lsevents.c](#)

## 6.5 `lsevents_queue_struct` Struct Reference

Storage definition for the events.

### Data Fields

- `char * evp`  
*name of the event*

#### 6.5.1 Detailed Description

Storage definition for the events.

Just a string for now. Perhaps one day we'll succumb to the temptation to add an argument or two.

Definition at line 17 of file lsevents.c.

#### 6.5.2 Field Documentation

##### 6.5.2.1 `char* lsevents_queue_struct::evp`

name of the event

Definition at line 18 of file lsevents.c.

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

- [lsevents.c](#)

## 6.6 `lslogging_queue_struct` Struct Reference

Our log object: time and message.

### Data Fields

- `struct timespec ltime`  
*time stamp: set when queued*
- `char lmsg [LSLOGGING_MSG_LENGTH]`  
*our message, truncated if too long*

### 6.6.1 Detailed Description

Our log object: time and message.

Definition at line 24 of file `lslogging.c`.

### 6.6.2 Field Documentation

#### 6.6.2.1 `char lslogging_queue_struct::lmsg[LSLOGGING_MSG_LENGTH]`

our message, truncated if too long

Definition at line 26 of file `lslogging.c`.

#### 6.6.2.2 `struct timespec lslogging_queue_struct::ltime`

time stamp: set when queued

Definition at line 25 of file `lslogging.c`.

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

- [lslogging.c](#)

## 6.7 `lspg_demandairrights_struct` Struct Reference

```
#include <pgpmac.h>
```

### Data Fields

- `pthread_mutex_t mutex`
- `pthread_cond_t cond`
- `int new_value_ready`

### 6.7.1 Detailed Description

Definition at line 200 of file `pgpmac.h`.

### 6.7.2 Field Documentation

#### 6.7.2.1 `pthread_cond_t lspg_demandairrights_struct::cond`

Definition at line 202 of file `pgpmac.h`.

#### 6.7.2.2 `pthread_mutex_t lspg_demandairrights_struct::mutex`

Definition at line 201 of file `pgpmac.h`.

#### 6.7.2.3 `int lspg_demandairrights_struct::new_value_ready`

Definition at line 203 of file `pgpmac.h`.

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

- `pgpmac.h`

## 6.8 `lspg_getcenter_struct` Struct Reference

Storage for getcenter query Used for the md2 ROTATE command that generates the centering movies.

```
#include <pgpmac.h>
```

### Data Fields

- `pthread_mutex_t mutex`  
*don't let the threads collide!*
- `pthread_cond_t cond`  
*provides signaling for when the query is done*
- `int new_value_ready`  
*used with condition*
- `int no_rows_returned`  
*flag in case no centering information was forthcoming*
- `int zoom`  
*the next zoom level to go to before taking the next movie*
- `int zoom_isnull`
- `double dcx`  
*center x change*
- `int dcx_isnull`
- `double dcy`  
*center y change*
- `int dcy_isnull`
- `double dax`  
*alignment x change*
- `int dax_isnull`
- `double day`  
*alignment y change*
- `int day_isnull`
- `double daz`  
*alignment z change*
- `int daz_isnull`

### 6.8.1 Detailed Description

Storage for getcenter query Used for the md2 ROTATE command that generates the centering movies.

Definition at line 214 of file pgpmac.h.

### 6.8.2 Field Documentation

#### 6.8.2.1 `pthread_cond_t lspg_getcenter_struct::cond`

provides signaling for when the query is done

Definition at line 216 of file pgpmac.h.

**6.8.2.2 double lspg\_getcenter\_struct::dax**

alignment x change

Definition at line 229 of file pgpmac.h.

**6.8.2.3 int lspg\_getcenter\_struct::dax\_isnull**

Definition at line 230 of file pgpmac.h.

**6.8.2.4 double lspg\_getcenter\_struct::day**

alignment y change

Definition at line 232 of file pgpmac.h.

**6.8.2.5 int lspg\_getcenter\_struct::day\_isnull**

Definition at line 233 of file pgpmac.h.

**6.8.2.6 double lspg\_getcenter\_struct::daz**

alignment z change

Definition at line 235 of file pgpmac.h.

**6.8.2.7 int lspg\_getcenter\_struct::daz\_isnull**

Definition at line 236 of file pgpmac.h.

**6.8.2.8 double lspg\_getcenter\_struct::dcx**

center x change

Definition at line 223 of file pgpmac.h.

**6.8.2.9 int lspg\_getcenter\_struct::dcx\_isnull**

Definition at line 224 of file pgpmac.h.

**6.8.2.10 double lspg\_getcenter\_struct::dcy**

center y change

Definition at line 226 of file pgpmac.h.

**6.8.2.11 int lspg\_getcenter\_struct::dcy\_isnull**

Definition at line 227 of file pgpmac.h.

#### 6.8.2.12 `pthread_mutex_t lsgc_getcenter_struct::mutex`

don't let the threads collide!

Definition at line 215 of file pgpmac.h.

#### 6.8.2.13 `int lsgc_getcenter_struct::new_value_ready`

used with condition

Definition at line 217 of file pgpmac.h.

#### 6.8.2.14 `int lsgc_getcenter_struct::no_rows_returned`

flag in case no centering information was forthcoming

Definition at line 218 of file pgpmac.h.

#### 6.8.2.15 `int lsgc_getcenter_struct::zoom`

the next zoom level to go to before taking the next movie

Definition at line 220 of file pgpmac.h.

#### 6.8.2.16 `int lsgc_getcenter_struct::zoom_isnull`

Definition at line 221 of file pgpmac.h.

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

- [pgpmac.h](#)

## 6.9 `lsgc_getcurrentsampleid_struct` Struct Reference

```
#include <pgpmac.h>
```

### Data Fields

- `pthread_mutex_t mutex`  
*practice safe threading*
- `pthread_cond_t cond`  
*for signaling*
- `int no_rows_returned`  
*flag for an empty return*
- `int new_value_ready`  
*OK, there is never a value, we need a variable for the conditional wait and this is what we call it everywhere else.*
- `unsigned int getcurrentsampleid`  
*the sample we think is mounted on the diffractometer*
- `int getcurrentsampleid_isnull`  
*the sample we think is mounted on the diffractometer*

### 6.9.1 Detailed Description

Definition at line 188 of file pgpmac.h.

### 6.9.2 Field Documentation

#### 6.9.2.1 `pthread_cond_t lspg_getcurrentsampleid_struct::cond`

for signaling

Definition at line 190 of file pgpmac.h.

#### 6.9.2.2 `unsigned int lspg_getcurrentsampleid_struct::getcurrentsampleid`

the sample we think is mounted on the diffractometer

Definition at line 193 of file pgpmac.h.

#### 6.9.2.3 `int lspg_getcurrentsampleid_struct::getcurrentsampleid_isnull`

the sample we think is mounted on the diffractometer

Definition at line 194 of file pgpmac.h.

#### 6.9.2.4 `pthread_mutex_t lspg_getcurrentsampleid_struct::mutex`

practice safe threading

Definition at line 189 of file pgpmac.h.

#### 6.9.2.5 `int lspg_getcurrentsampleid_struct::new_value_ready`

OK, there is never a value, we need a variable for the conditional wait and this is what we call it everywhere else.

Definition at line 192 of file pgpmac.h.

#### 6.9.2.6 `int lspg_getcurrentsampleid_struct::no_rows_returned`

flag for an empty return

Definition at line 191 of file pgpmac.h.

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

- [pgpmac.h](#)

## 6.10 `lspg_lock_detector_struct` Struct Reference

lock detector object Implements detector lock for exposure control

### Data Fields

- `pthread_mutex_t mutex`
- `pthread_cond_t cond`
- `int new_value_ready`

### 6.10.1 Detailed Description

lock detector object Implements detector lock for exposure control

Definition at line 1065 of file [lspg.c](#).

### 6.10.2 Field Documentation

#### 6.10.2.1 `pthread_cond_t lspg_lock_detector_struct::cond`

Definition at line 1067 of file [lspg.c](#).

#### 6.10.2.2 `pthread_mutex_t lspg_lock_detector_struct::mutex`

Definition at line 1066 of file [lspg.c](#).

#### 6.10.2.3 `int lspg_lock_detector_struct::new_value_ready`

Definition at line 1068 of file [lspg.c](#).

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

- [lspg.c](#)

## 6.11 `lspg_lock_diffractometer_struct` Struct Reference

Object used to implement locking the diffractometer Critical to exposure timing.

### Data Fields

- `pthread_mutex_t mutex`
- `pthread_cond_t cond`
- `int new_value_ready`

### 6.11.1 Detailed Description

Object used to implement locking the diffractometer Critical to exposure timing.

Definition at line 1006 of file [lspg.c](#).

### 6.11.2 Field Documentation

#### 6.11.2.1 `pthread_cond_t lspg_lock_diffractometer_struct::cond`

Definition at line 1008 of file [lspg.c](#).

#### 6.11.2.2 `pthread_mutex_t lspg_lock_diffractometer_struct::mutex`

Definition at line 1007 of file [lspg.c](#).

### 6.11.2.3 int lspg\_lock\_diffractometer\_struct::new\_value\_ready

Definition at line 1009 of file lspg.c.

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

- [lspg.c](#)

## 6.12 lspg\_nextsample\_struct Struct Reference

Returns the next sample number Just a 32 bit int (Hal, take that, nextshot!)

```
#include <pgpmac.h>
```

### Data Fields

- `pthread_mutex_t mutex`  
*Our mutex.*
- `pthread_cond_t cond`  
*Our condition.*
- `int new_value_ready`  
*flag for our condition*
- `int no_rows_returned`  
*just in case, though this query should always return an integer, perhaps 0*
- `unsigned int nextsample`  
*sample number (4 8-bit segments: station, dewar (lid), puck, and position in the puck)*
- `int nextsample_isnull`  
*shouldn't ever be set, but if we change the logic of this call in PG then we are ready for it here.*

### 6.12.1 Detailed Description

Returns the next sample number Just a 32 bit int (Hal, take that, nextshot!)

Definition at line 261 of file pgpmac.h.

### 6.12.2 Field Documentation

#### 6.12.2.1 pthread\_cond\_t lspg\_nextsample\_struct::cond

Our condition.

Definition at line 263 of file pgpmac.h.

#### 6.12.2.2 pthread\_mutex\_t lspg\_nextsample\_struct::mutex

Our mutex.

Definition at line 262 of file pgpmac.h.

#### 6.12.2.3 int lspg\_nextsample\_struct::new\_value\_ready

flag for our condition

Definition at line 264 of file pgpmac.h.

#### 6.12.2.4 unsigned int lspg\_nexsample\_struct::nexsample

sample number (4 8-bit segments: station, dewar (lid), puck, and position in the puck)

Definition at line 267 of file pgpmac.h.

#### 6.12.2.5 int lspg\_nexsample\_struct::nexsample\_isnull

shouldn't ever be set, but if we change the logic of this call in PG then we are ready for it here.

Definition at line 268 of file pgpmac.h.

#### 6.12.2.6 int lspg\_nexsample\_struct::no\_rows\_returned

just in case, though this query should always return an integer, perhaps 0

Definition at line 265 of file pgpmac.h.

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

- [pgpmac.h](#)

## 6.13 lspg\_nextshot\_struct Struct Reference

Storage definition for nextshot query.

```
#include <pgpmac.h>
```

### Data Fields

- `pthread_mutex_t mutex`  
*Our mutex for sanity in the multi-threaded program.*
- `pthread_cond_t cond`  
*Condition to wait for a response from our postgresql server.*
- `int new_value_ready`  
*Our flag for the condition to wait for.*
- `int no_rows_returned`  
*flag indicating that no rows were returned.*
- `char * dsdir`  
*Directory for data relative to the ESAF home directory.*
- `int dsdir_isnull`
- `char * dspid`  
*ID string identifying this dataset.*
- `int dspid_isnull`
- `double dsowidth`  
*dataset defined oscillation width*
- `int dsowidth_isnull`
- `char * dsoscaxis`  
*dataset defined oscillation axis (always omega)*
- `int dsoscaxis_isnull`
- `double dsexp`  
*dataset defined exposure time*
- `int dsexp_isnull`
- `long long skey`

*key identifying a particular image*

- int **skey\_isnull**
- double **sstart**  
*starting angle*
- int **sstart\_isnull**
- char \* **sfn**  
*file name*
- int **sfn\_isnull**
- double **dsphi**  
*dataset defined starting phi angle*
- int **dsphi\_isnull**
- double **dsomega**  
*dataset defined starting omega angle*
- int **dsomega\_isnull**
- double **dskappa**  
*dataset defined starting kappa angle*
- int **dskappa\_isnull**
- double **dsdist**  
*dataset defined detector distance*
- int **dsdist\_isnull**
- double **dsnrg**  
*dataset defined energy*
- int **dsnrg\_isnull**
- unsigned int **dshpid**  
*sample holder ID*
- int **dshpid\_isnull**
- double **cx**  
*centering table x position*
- int **cx\_isnull**
- double **cy**  
*centering table y position*
- int **cy\_isnull**
- double **ax**  
*alignment table x position*
- int **ax\_isnull**
- double **ay**  
*alignment table y position*
- int **ay\_isnull**
- double **az**  
*alignment table z position*
- int **az\_isnull**
- int **active**  
*flag: 1=move to indicated center position, 0=don't move center or alignment tables*
- int **active\_isnull**
- int **sindex**  
*index of frame (used to generate the file extension)*
- int **sindex\_isnull**
- char \* **stype**  
*"Normal" or "Gridsearch"*
- int **stype\_isnull**
- double **dsowidth2**  
*next image oscillation width*

- int `dsowidth2_isnull`
- char \* `dsoscaxis2`

*next image oscillation axis (always "omega")*
- int `dsoscaxis2_isnull`
- double `dsexp2`

*next image exposure time*
- int `dsexp2_isnull`
- double `sstart2`

*next image start angle*
- int `sstart2_isnull`
- double `dsphi2`

*next image phi position*
- int `dsphi2_isnull`
- double `dsomega2`

*next image omega position*
- int `dsomega2_isnull`
- double `dskappa2`

*next image kappa position*
- int `dskappa2_isnull`
- double `dsdist2`

*next image distance*
- int `dsdist2_isnull`
- double `dsnrg2`

*next image energy*
- int `dsnrg2_isnull`
- double `cx2`

*next image centering table x position*
- int `cx2_isnull`
- double `cy2`

*next image centering table y position*
- int `cy2_isnull`
- double `ax2`

*next image alignment x position*
- int `ax2_isnull`
- double `ay2`

*next image alignment y position*
- int `ay2_isnull`
- double `az2`

*next image alignment z position*
- int `az2_isnull`
- int `active2`

*flag: 1 if next image should use the above centering parameters*
- int `active2_isnull`
- int `sindex2`

*next image index number*
- int `sindex2_isnull`
- char \* `stype2`

*next image type ("Normal" or "Gridsearch")*
- int `stype2_isnull`

### 6.13.1 Detailed Description

Storage definition for nextshot query.

The next shot query returns all the information needed to collect the next data frame. Since SQL allows for null fields independently from blank strings a separate integer is used as a flag for this case. This adds to the program complexity but allows for some important cases. Suck it up.

Definition at line 281 of file pgpmac.h.

### 6.13.2 Field Documentation

#### 6.13.2.1 `int lspg_nextshot_struct::active`

flag: 1=move to indicated center position, 0=don't move center or alignment tables

Definition at line 344 of file pgpmac.h.

#### 6.13.2.2 `int lspg_nextshot_struct::active2`

flag: 1 if next image should use the above centering parameters

Definition at line 395 of file pgpmac.h.

#### 6.13.2.3 `int lspg_nextshot_struct::active2_isnull`

Definition at line 396 of file pgpmac.h.

#### 6.13.2.4 `int lspg_nextshot_struct::active_isnull`

Definition at line 345 of file pgpmac.h.

#### 6.13.2.5 `double lspg_nextshot_struct::ax`

alignment table x position

Definition at line 335 of file pgpmac.h.

#### 6.13.2.6 `double lspg_nextshot_struct::ax2`

next image alignment x position

Definition at line 386 of file pgpmac.h.

#### 6.13.2.7 `int lspg_nextshot_struct::ax2_isnull`

Definition at line 387 of file pgpmac.h.

#### 6.13.2.8 `int lspg_nextshot_struct::ax_isnull`

Definition at line 336 of file pgpmac.h.

**6.13.2.9 double lsgp\_nextshot\_struct::ay**

alignment table y position

Definition at line 338 of file pgpmac.h.

**6.13.2.10 double lsgp\_nextshot\_struct::ay2**

next image alignment y position

Definition at line 389 of file pgpmac.h.

**6.13.2.11 int lsgp\_nextshot\_struct::ay2\_isnull**

Definition at line 390 of file pgpmac.h.

**6.13.2.12 int lsgp\_nextshot\_struct::ay\_isnull**

Definition at line 339 of file pgpmac.h.

**6.13.2.13 double lsgp\_nextshot\_struct::az**

alignment table z position

Definition at line 341 of file pgpmac.h.

**6.13.2.14 double lsgp\_nextshot\_struct::az2**

next image alignment z position

Definition at line 392 of file pgpmac.h.

**6.13.2.15 int lsgp\_nextshot\_struct::az2\_isnull**

Definition at line 393 of file pgpmac.h.

**6.13.2.16 int lsgp\_nextshot\_struct::az\_isnull**

Definition at line 342 of file pgpmac.h.

**6.13.2.17 pthread\_cond\_t lsgp\_nextshot\_struct::cond**

Condition to wait for a response from our postgresql server.

Definition at line 283 of file pgpmac.h.

**6.13.2.18 double lsgp\_nextshot\_struct::cx**

centering table x position

Definition at line 329 of file pgpmac.h.

6.13.2.19 double **lspg\_nextshot\_struct::cx2**

next image centering table x position

Definition at line 380 of file pgpmac.h.

6.13.2.20 int **lspg\_nextshot\_struct::cx2\_isnull**

Definition at line 381 of file pgpmac.h.

6.13.2.21 int **lspg\_nextshot\_struct::cx\_isnull**

Definition at line 330 of file pgpmac.h.

6.13.2.22 double **lspg\_nextshot\_struct::cy**

centering table y position

Definition at line 332 of file pgpmac.h.

6.13.2.23 double **lspg\_nextshot\_struct::cy2**

next image centering table y position

Definition at line 383 of file pgpmac.h.

6.13.2.24 int **lspg\_nextshot\_struct::cy2\_isnull**

Definition at line 384 of file pgpmac.h.

6.13.2.25 int **lspg\_nextshot\_struct::cy\_isnull**

Definition at line 333 of file pgpmac.h.

6.13.2.26 char\* **lspg\_nextshot\_struct::dsdir**

Directory for data relative to the ESAF home directory.

Definition at line 287 of file pgpmac.h.

6.13.2.27 int **lspg\_nextshot\_struct::dsdir\_isnull**

Definition at line 288 of file pgpmac.h.

6.13.2.28 double **lspg\_nextshot\_struct::dsdist**

dataset defined detector distance

Definition at line 320 of file pgpmac.h.

6.13.2.29 double lsgp\_nextshot\_struct::dsdist2

next image distance

Definition at line 374 of file pgpmac.h.

6.13.2.30 int lsgp\_nextshot\_struct::dsdist2\_isnull

Definition at line 375 of file pgpmac.h.

6.13.2.31 int lsgp\_nextshot\_struct::dsdist\_isnull

Definition at line 321 of file pgpmac.h.

6.13.2.32 double lsgp\_nextshot\_struct::dsexp

dataset defined exposure time

Definition at line 299 of file pgpmac.h.

6.13.2.33 double lsgp\_nextshot\_struct::dsexp2

next image exposure time

Definition at line 359 of file pgpmac.h.

6.13.2.34 int lsgp\_nextshot\_struct::dsexp2\_isnull

Definition at line 360 of file pgpmac.h.

6.13.2.35 int lsgp\_nextshot\_struct::dsexp\_isnull

Definition at line 300 of file pgpmac.h.

6.13.2.36 unsigned int lsgp\_nextshot\_struct::dshpid

sample holder ID

Definition at line 326 of file pgpmac.h.

6.13.2.37 int lsgp\_nextshot\_struct::dshpid\_isnull

Definition at line 327 of file pgpmac.h.

6.13.2.38 double lsgp\_nextshot\_struct::dskappa

dataset defined starting kappa angle

Definition at line 317 of file pgpmac.h.

6.13.2.39 double **lspg\_nextshot\_struct::dskappa2**

next image kappa position

Definition at line 371 of file pgpmac.h.

6.13.2.40 int **lspg\_nextshot\_struct::dskappa2\_isnull**

Definition at line 372 of file pgpmac.h.

6.13.2.41 int **lspg\_nextshot\_struct::dskappa\_isnull**

Definition at line 318 of file pgpmac.h.

6.13.2.42 double **lspg\_nextshot\_struct::dsnrg**

dataset defined energy

Definition at line 323 of file pgpmac.h.

6.13.2.43 double **lspg\_nextshot\_struct::dsnrg2**

next image energy

Definition at line 377 of file pgpmac.h.

6.13.2.44 int **lspg\_nextshot\_struct::dsnrg2\_isnull**

Definition at line 378 of file pgpmac.h.

6.13.2.45 int **lspg\_nextshot\_struct::dsnrg\_isnull**

Definition at line 324 of file pgpmac.h.

6.13.2.46 double **lspg\_nextshot\_struct::dsomega**

dataset defined starting omega angle

Definition at line 314 of file pgpmac.h.

6.13.2.47 double **lspg\_nextshot\_struct::dsomega2**

next image omega position

Definition at line 368 of file pgpmac.h.

6.13.2.48 int **lspg\_nextshot\_struct::dsomega2\_isnull**

Definition at line 369 of file pgpmac.h.

6.13.2.49 int **lspg\_nextshot\_struct::dsomega\_isnull**

Definition at line 315 of file pgpmac.h.

**6.13.2.50 char\* lsgp\_nextshot\_struct::dsoscaxis**

dataset defined oscillation axis (always omega)

Definition at line 296 of file pgpmac.h.

**6.13.2.51 char\* lsgp\_nextshot\_struct::dsoscaxis2**

next image oscillation axis (always "omega")

Definition at line 356 of file pgpmac.h.

**6.13.2.52 int lsgp\_nextshot\_struct::dsoscaxis2\_isnull**

Definition at line 357 of file pgpmac.h.

**6.13.2.53 int lsgp\_nextshot\_struct::dsoscaxis\_isnull**

Definition at line 297 of file pgpmac.h.

**6.13.2.54 double lsgp\_nextshot\_struct::dsowidth**

dataset defined oscillation width

Definition at line 293 of file pgpmac.h.

**6.13.2.55 double lsgp\_nextshot\_struct::dsowidth2**

next image oscillation width

Definition at line 353 of file pgpmac.h.

**6.13.2.56 int lsgp\_nextshot\_struct::dsowidth2\_isnull**

Definition at line 354 of file pgpmac.h.

**6.13.2.57 int lsgp\_nextshot\_struct::dsowidth\_isnull**

Definition at line 294 of file pgpmac.h.

**6.13.2.58 double lsgp\_nextshot\_struct::dsphi**

dataset defined starting phi angle

Definition at line 311 of file pgpmac.h.

**6.13.2.59 double lsgp\_nextshot\_struct::dsphi2**

next image phi position

Definition at line 365 of file pgpmac.h.

6.13.2.60 int lspg\_nextshot\_struct::dsphi2\_isnull

Definition at line 366 of file pgpmac.h.

6.13.2.61 int lspg\_nextshot\_struct::dsphi\_isnull

Definition at line 312 of file pgpmac.h.

6.13.2.62 char\* lspg\_nextshot\_struct::dspid

ID string identifying this dataset.

Definition at line 290 of file pgpmac.h.

6.13.2.63 int lspg\_nextshot\_struct::dspid\_isnull

Definition at line 291 of file pgpmac.h.

6.13.2.64 pthread\_mutex\_t lspg\_nextshot\_struct::mutex

Our mutex for sanity in the multi-threaded program.

Definition at line 282 of file pgpmac.h.

6.13.2.65 int lspg\_nextshot\_struct::new\_value\_ready

Our flag for the condition to wait for.

Definition at line 284 of file pgpmac.h.

6.13.2.66 int lspg\_nextshot\_struct::no\_rows\_returned

flag indicating that no rows were returned.

Definition at line 285 of file pgpmac.h.

6.13.2.67 char\* lspg\_nextshot\_struct::sfn

file name

Definition at line 308 of file pgpmac.h.

6.13.2.68 int lspg\_nextshot\_struct::sfn\_isnull

Definition at line 309 of file pgpmac.h.

6.13.2.69 int lspg\_nextshot\_struct::sindex

index of frame (used to generate the file extension)

Definition at line 347 of file pgpmac.h.

**6.13.2.70 int lsgp\_nextshot\_struct::sindex2**

next image index number

Definition at line 398 of file pgpmac.h.

**6.13.2.71 int lsgp\_nextshot\_struct::sindex2\_isnull**

Definition at line 399 of file pgpmac.h.

**6.13.2.72 int lsgp\_nextshot\_struct::sindex\_isnull**

Definition at line 348 of file pgpmac.h.

**6.13.2.73 long long lsgp\_nextshot\_struct::skey**

key identifying a particular image

Definition at line 302 of file pgpmac.h.

**6.13.2.74 int lsgp\_nextshot\_struct::skey\_isnull**

Definition at line 303 of file pgpmac.h.

**6.13.2.75 double lsgp\_nextshot\_struct::sstart**

starting angle

Definition at line 305 of file pgpmac.h.

**6.13.2.76 double lsgp\_nextshot\_struct::sstart2**

next image start angle

Definition at line 362 of file pgpmac.h.

**6.13.2.77 int lsgp\_nextshot\_struct::sstart2\_isnull**

Definition at line 363 of file pgpmac.h.

**6.13.2.78 int lsgp\_nextshot\_struct::sstart\_isnull**

Definition at line 306 of file pgpmac.h.

**6.13.2.79 char\* lsgp\_nextshot\_struct::stype**

"Normal" or "Gridsearch"

Definition at line 350 of file pgpmac.h.

6.13.2.80 `char* lspg_nextshot_struct::stype2`

next image type ("Normal" or "Gridsearch")

Definition at line 401 of file pgpmac.h.

6.13.2.81 `int lspg_nextshot_struct::stype2_isnull`

Definition at line 402 of file pgpmac.h.

6.13.2.82 `int lspg_nextshot_struct::stype_isnull`

Definition at line 351 of file pgpmac.h.

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

- [pgpmac.h](#)

## 6.14 `lspg_seq_run_prep_struct` Struct Reference

Data collection running object.

### Data Fields

- `pthread_mutex_t mutex`
- `pthread_cond_t cond`
- `int new_value_ready`

### 6.14.1 Detailed Description

Data collection running object.

Definition at line 1123 of file lspg.c.

### 6.14.2 Field Documentation

6.14.2.1 `pthread_cond_t lspg_seq_run_prep_struct::cond`

Definition at line 1125 of file lspg.c.

6.14.2.2 `pthread_mutex_t lspg_seq_run_prep_struct::mutex`

Definition at line 1124 of file lspg.c.

6.14.2.3 `int lspg_seq_run_prep_struct::new_value_ready`

Definition at line 1126 of file lspg.c.

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

- [lspg.c](#)

## 6.15 lspg\_starttransfer\_struct Struct Reference

returns 1 if transfer can continue 0 to abort

```
#include <pgpmac.h>
```

### Data Fields

- `pthread_mutex_t mutex`  
*Our mutex.*
- `pthread_cond_t cond`  
*Our condition.*
- `int new_value_ready`  
*flag for our condition*
- `int no_rows_returned`  
*just in case, though this query should always return an integer, perhaps 0*
- `unsigned int starttransfer`  
*sample number (4 8-bit segments: station, dewar (lid), puck, and position in the puck)*

### 6.15.1 Detailed Description

returns 1 if transfer can continue 0 to abort

Definition at line 247 of file pgpmac.h.

### 6.15.2 Field Documentation

#### 6.15.2.1 `pthread_cond_t lspg_starttransfer_struct::cond`

Our condition.

Definition at line 249 of file pgpmac.h.

#### 6.15.2.2 `pthread_mutex_t lspg_starttransfer_struct::mutex`

Our mutex.

Definition at line 248 of file pgpmac.h.

#### 6.15.2.3 `int lspg_starttransfer_struct::new_value_ready`

flag for our condition

Definition at line 250 of file pgpmac.h.

#### 6.15.2.4 `int lspg_starttransfer_struct::no_rows_returned`

just in case, though this query should always return an integer, perhaps 0

Definition at line 251 of file pgpmac.h.

### 6.15.2.5 unsigned int lspg\_starttransfer\_struct::starttransfer

sample number (4 8-bit segments: station, dewar (lid), puck, and position in the puck)

Definition at line 253 of file pgpmac.h.

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

- [pgpmac.h](#)

## 6.16 lspg\_wait\_for\_detector\_struct Struct Reference

Object that implements detector / spindle timing We use database locks for exposure control and this implements the md2 portion of this handshake.

### Data Fields

- `pthread_mutex_t mutex`
- `pthread_cond_t cond`
- `int new_value_ready`

### 6.16.1 Detailed Description

Object that implements detector / spindle timing We use database locks for exposure control and this implements the md2 portion of this handshake.

Definition at line 941 of file lspg.c.

### 6.16.2 Field Documentation

#### 6.16.2.1 pthread\_cond\_t lspg\_wait\_for\_detector\_struct::cond

Definition at line 943 of file lspg.c.

#### 6.16.2.2 pthread\_mutex\_t lspg\_wait\_for\_detector\_struct::mutex

Definition at line 942 of file lspg.c.

#### 6.16.2.3 int lspg\_wait\_for\_detector\_struct::new\_value\_ready

Definition at line 944 of file lspg.c.

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

- [lspg.c](#)

## 6.17 lspg\_waitcryo\_struct Struct Reference

```
#include <pgpmac.h>
```

## Data Fields

- `pthread_mutex_t mutex`  
*practice safe threading*
- `pthread_cond_t cond`  
*for signaling*
- `int new_value_ready`  
*OK, there is never a value, we need a variable for the conditional wait and this is what we call it everywhere else.*

### 6.17.1 Detailed Description

Definition at line 180 of file pgpmac.h.

### 6.17.2 Field Documentation

#### 6.17.2.1 `pthread_cond_t lsgp_waitcryo_struct::cond`

for signaling

Definition at line 182 of file pgpmac.h.

#### 6.17.2.2 `pthread_mutex_t lsgp_waitcryo_struct::mutex`

practice safe threading

Definition at line 181 of file pgpmac.h.

#### 6.17.2.3 `int lsgp_waitcryo_struct::new_value_ready`

OK, there is never a value, we need a variable for the conditional wait and this is what we call it everywhere else.

Definition at line 183 of file pgpmac.h.

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

- [pgpmac.h](#)

## 6.18 LsgpQueryQueueStruct Struct Reference

Store each query along with its callback function.

```
#include <pgpmac.h>
```

## Data Fields

- `char qs [LS_PG_QUERY_STRING_LENGTH]`  
*our queries should all be pretty short as we'll just be calling functions: fixed length here simplifies memory management*
- `void(* onResponse )(struct LsgpQueryQueueStruct *qq, PGresult *pgr)`  
*Callback function for when a query returns a result.*

### 6.18.1 Detailed Description

Store each query along with it's callback function.

All calls are asynchronous

Definition at line 175 of file pgpmac.h.

### 6.18.2 Field Documentation

#### 6.18.2.1 void(\* lspgQueryQueueStruct::onResponse)(struct lspgQueryQueueStruct \*qq, PGresult \*pgr)

Callback function for when a query returns a result.

Definition at line 177 of file pgpmac.h.

#### 6.18.2.2 char lspgQueryQueueStruct::qs[LS\_PG\_QUERY\_STRING\_LENGTH]

our queries should all be pretty short as we'll just be calling functions: fixed length here simplifies memory management

Definition at line 176 of file pgpmac.h.

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

- [pgpmac.h](#)

## 6.19 lspmac\_ascii\_buffers\_struct Struct Reference

### Data Fields

- uint16\_t [command\\_buf](#)
- uint16\_t [command\\_buf\\_cc](#)
- char [command\\_str](#) [160]
- uint16\_t [response\\_buf](#)
- uint16\_t [response\\_n](#)
- char [response\\_str](#) [256]

### 6.19.1 Detailed Description

Definition at line 357 of file lspmac.c.

### 6.19.2 Field Documentation

#### 6.19.2.1 uint16\_t lspmac\_ascii\_buffers\_struct::command\_buf

Definition at line 359 of file lspmac.c.

#### 6.19.2.2 uint16\_t lspmac\_ascii\_buffers\_struct::command\_buf\_cc

Definition at line 360 of file lspmac.c.

### 6.19.2.3 `char lspmacc_ascii_buffers_struct::command_str[160]`

Definition at line 361 of file lspmacc.c.

### 6.19.2.4 `uint16_t lspmacc_ascii_buffers_struct::response_buf`

Definition at line 362 of file lspmacc.c.

### 6.19.2.5 `uint16_t lspmacc_ascii_buffers_struct::response_n`

Definition at line 363 of file lspmacc.c.

### 6.19.2.6 `char lspmacc_ascii_buffers_struct::response_str[256]`

Definition at line 364 of file lspmacc.c.

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

- [lspmacc.c](#)

## 6.20 `lspmac.bi_struct` Struct Reference

Storage for binary inputs.

```
#include <pgpmac.h>
```

### Data Fields

- `int * ptr`  
*points to the location in the status buffer*
- `pthread_mutex_t mutex`  
*so we don't get confused*
- `int mask`  
*mask for the bit in the status register*
- `int position`  
*the current value.*
- `int previous`  
*the previous value*
- `int first_time`  
*flag indicating we've not read the input even once*
- `char * changeEventOn`  
*Event to send when the value changes to 1.*
- `char * changeEventOff`  
*Event to send when the value changes to 0.*

### 6.20.1 Detailed Description

Storage for binary inputs.

Definition at line 160 of file pgpmac.h.

## 6.20.2 Field Documentation

### 6.20.2.1 **char\* lspmac.bi\_struct::changeEventOff**

Event to send when the value changes to 0.

Definition at line 168 of file pgpmac.h.

### 6.20.2.2 **char\* lspmac.bi\_struct::changeEventOn**

Event to send when the value changes to 1.

Definition at line 167 of file pgpmac.h.

### 6.20.2.3 **int lspmac.bi\_struct::first\_time**

flag indicating we've not read the input even once

Definition at line 166 of file pgpmac.h.

### 6.20.2.4 **int lspmac.bi\_struct::mask**

mask for the bit in the status register

Definition at line 163 of file pgpmac.h.

### 6.20.2.5 **pthread\_mutex\_t lspmac.bi\_struct::mutex**

so we don't get confused

Definition at line 162 of file pgpmac.h.

### 6.20.2.6 **int lspmac.bi\_struct::position**

the current value.

Definition at line 164 of file pgpmac.h.

### 6.20.2.7 **int lspmac.bi\_struct::previous**

the previous value

Definition at line 165 of file pgpmac.h.

### 6.20.2.8 **int\* lspmac.bi\_struct::ptr**

points to the location in the status buffer

Definition at line 161 of file pgpmac.h.

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

- [pgpmac.h](#)

## 6.21 `lspmac_cmd_queue_struct` Struct Reference

PMAC command queue item.

```
#include <pgpmac.h>
```

### Data Fields

- `pmac_cmd_t pcmd`  
*the pmac command to send*
- `int no_reply`  
*1 = no reply is expected, 0 = expect a reply*
- `struct timespec time_sent`  
*time this item was dequeued and sent to the pmac*
- `char * event`  
*event name to send*
- `void(* onResponse )(struct lspmac_cmd_queue_struct *, int, char *)`  
*function to call when response is received. args are (int fd, nreturned, buffer)*

### 6.21.1 Detailed Description

PMAC command queue item.

Command queue items are fixed length to simplify memory management.

Definition at line 86 of file pgpmac.h.

### 6.21.2 Field Documentation

#### 6.21.2.1 `char* lspmac_cmd_queue_struct::event`

event name to send

Definition at line 90 of file pgpmac.h.

#### 6.21.2.2 `int lspmac_cmd_queue_struct::no_reply`

`1 = no reply is expected, 0 = expect a reply`

Definition at line 88 of file pgpmac.h.

#### 6.21.2.3 `void(* lspmac_cmd_queue_struct::onResponse)(struct lspmac_cmd_queue_struct *, int, char *)`

function to call when response is received. args are (int fd, nreturned, buffer)

Definition at line 91 of file pgpmac.h.

#### 6.21.2.4 `pmac_cmd_t lspmac_cmd_queue_struct::pcmd`

the pmac command to send

Definition at line 87 of file pgpmac.h.

## 6.21.2.5 struct timespec lsppmac\_cmd\_queue\_struct::time\_sent

time this item was dequeued and sent to the pmac

Definition at line 89 of file pgpmac.h.

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

- [pgpmac.h](#)

## 6.22 lsppmac\_combined\_move\_struct Struct Reference

### Data Fields

- int [Delta](#)
- int [moveme](#)
- int [coord\\_num](#)
- int [axis](#)

### 6.22.1 Detailed Description

Definition at line 380 of file lsppmac.c.

### 6.22.2 Field Documentation

#### 6.22.2.1 int lsppmac\_combined\_move\_struct::axis

Definition at line 384 of file lsppmac.c.

#### 6.22.2.2 int lsppmac\_combined\_move\_struct::coord\_num

Definition at line 383 of file lsppmac.c.

#### 6.22.2.3 int lsppmac\_combined\_move\_struct::Delta

Definition at line 381 of file lsppmac.c.

#### 6.22.2.4 int lsppmac\_combined\_move\_struct::moveme

Definition at line 382 of file lsppmac.c.

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

- [lsppmac.c](#)

## 6.23 lsppmac\_dpascii\_queue\_struct Struct Reference

### Data Fields

- char \* [event](#)
- char [pl](#)[160]

### 6.23.1 Detailed Description

Definition at line 371 of file lspmac.c.

### 6.23.2 Field Documentation

#### 6.23.2.1 `char* lspmac_dpascii_queue_struct::event`

Definition at line 372 of file lspmac.c.

#### 6.23.2.2 `char lspmac_dpascii_queue_struct::pl[160]`

Definition at line 373 of file lspmac.c.

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

- [lspmac.c](#)

## 6.24 `lspmac_motor_struct` Struct Reference

Motor information.

```
#include <pgpmac.h>
```

### Data Fields

- int `magic`  
*magic number identifying this as a motor structure*
- `pthread_mutex_t mutex`  
*coordinate waiting for motor to be done*
- `pthread_cond_t cond`  
*used to signal when a motor is done moving*
- int `not_done`  
*set to 1 when request is queued, zero after motion has toggled*
- `void(* read )(struct lspmac_motor_struct *)`  
*method to read the motor status and position*
- int `command_sent`  
*Motion command verified sent to pmac.*
- int `motion_seen`  
*set to 1 when motion has been verified to have started*
- `pmac_cmd_queue_t * pq`  
*the queue item requesting motion. Used to check time request was made*
- int `homming`  
*Homing routine started.*
- int `requested_pos_cnts`  
*requested position*
- `int * actual_pos_cnts_p`  
*pointer to the md2\_status structure to the actual position*
- int `actual_pos_cnts`  
*local copy of actual counts so only our mutex is needed to read*
- double `position`

- double **reported\_pg\_position**  
*previous position reported to postgresql*
  - double **reported\_position**  
*previous position reported to redis*
  - double **requested\_position**  
*The position as requested by the user.*
  - int \* **status1\_p**  
*First 24 bit PMAC motor status word.*
  - int **status1**  
*local copy of status1*
  - int \* **status2\_p**  
*Sectond 24 bit PMAC motor status word.*
  - int **status2**  
*local copy of status2*
  - char \* **dac\_mvar**  
*controlling mvariable as a string*
  - char \* **name**  
*Name of motor as refered by ls database kvs table.*
  - **lsredis\_obj\_t \* active**  
*Use the motor ("true") or not ("false")*
  - **lsredis\_obj\_t \* active\_init**  
*pmac commands to make this motor active*
  - **lsredis\_obj\_t \* axis**  
*the axis (X, Y, Z, etc) or null if not in a coordinate system*
  - **lsredis\_obj\_t \* coord\_num**  
*coordinate system this motor belongs to (0 if none)*
  - **lsredis\_obj\_t \* home**  
*pmac commands to home motor*
  - **lsredis\_obj\_t \* inactive\_init**  
*pmac commands to inactivate the motor*
  - **lsredis\_obj\_t \* in\_position\_band**  
*moves within this amount are ignored UNITS ARE 1/16 COUNT*
  - **lsredis\_obj\_t \* max\_accel**  
*our maximum acceleration (cts/msec<sup>2</sup>)*
  - **lsredis\_obj\_t \* max\_pos**  
*our maximum position (soft limit)*
  - **lsredis\_obj\_t \* max\_speed**  
*our maximum speed (cts/msec)*
  - **lsredis\_obj\_t \* min\_pos**  
*our minimum position (soft limit)*
  - **lsredis\_obj\_t \* motor\_num**  
*pmac motor number*
  - **lsredis\_obj\_t \* neutral\_pos**  
*zero offset*
  - **lsredis\_obj\_t \* pos\_limit\_hit**  
*positive limit status*
  - **lsredis\_obj\_t \* neg\_limit\_hit**  
*negative limit status*
  - **lsredis\_obj\_t \* precision**  
*moves of less than this amount may be ignored*

- `Isredis_obj_t * printf_fmt`  
*printf format*
- `Isredis_obj_t * redis_fmt`  
*special format string to create text array for putting the position back into redis*
- `Isredis_obj_t * redis_position`  
*how we report our position to the world*
- `Isredis_obj_t * status_str`  
*A talky version of the status.*
- `Isredis_obj_t * u2c`  
*conversion from counts to units: 0.0 means not loaded yet*
- `Isredis_obj_t * unit`  
*string to use as the units*
- `Isredis_obj_t * update_resolution`  
*Change needs to be at least this big to report as a new position to the database.*
- `char * write_fmt`  
*Format string to write requested position to PMAC used for binary i/o.*
- `int * read_ptr`  
*With read\_mask finds bit to read for binary i/o.*
- `int read_mask`  
*With read\_ptr find bit to read for binary i/o.*
- `int(* moveAbs )(struct lspmac_motor_struct *, double)`  
*function to move the motor*
- `int(* jogAbs )(struct lspmac_motor_struct *, double)`  
*function to move the motor*
- `double * lut`  
*lookup table (instead of u2c)*
- `int nlut`  
*length of lut*
- `WINDOW * win`  
*our ncurses window*

### 6.24.1 Detailed Description

Motor information.

A catchall for motors and motor like objects. Not all members are used by all objects.

Definition at line 101 of file pgpmac.h.

### 6.24.2 Field Documentation

#### 6.24.2.1 `Isredis_obj_t* lspmac_motor_struct::active`

Use the motor ("true") or not ("false")

Definition at line 124 of file pgpmac.h.

#### 6.24.2.2 `Isredis_obj_t* lspmac_motor_struct::active_init`

pmac commands to make this motor active

Definition at line 125 of file pgpmac.h.

**6.24.2.3 int lspmac\_motor\_struct::actual\_pos\_cnts**

local copy of actual counts so only our mutex is needed to read

Definition at line 113 of file pgpmac.h.

**6.24.2.4 int\* lspmac\_motor\_struct::actual\_pos\_cnts\_p**

pointer to the md2\_status structure to the actual position

Definition at line 112 of file pgpmac.h.

**6.24.2.5 Isredis\_obj\_t\* lspmac\_motor\_struct::axis**

the axis (X, Y, Z, etc) or null if not in a coordinate system

Definition at line 126 of file pgpmac.h.

**6.24.2.6 int lspmac\_motor\_struct::command\_sent**

Motion command verified sent to pmac.

Definition at line 107 of file pgpmac.h.

**6.24.2.7 pthread\_cond\_t lspmac\_motor\_struct::cond**

used to signal when a motor is done moving

Definition at line 104 of file pgpmac.h.

**6.24.2.8 Isredis\_obj\_t\* lspmac\_motor\_struct::coord\_num**

coordinate system this motor belongs to (0 if none)

Definition at line 127 of file pgpmac.h.

**6.24.2.9 char\* lspmac\_motor\_struct::dac\_mvar**

controlling mvariable as a string

Definition at line 122 of file pgpmac.h.

**6.24.2.10 Isredis\_obj\_t\* lspmac\_motor\_struct::home**

pmac commands to home motor

Definition at line 128 of file pgpmac.h.

**6.24.2.11 int lspmac\_motor\_struct::homing**

Homing routine started.

Definition at line 110 of file pgpmac.h.

**6.24.2.12 `Isredis_obj_t* lspmac_motor_struct::in_position_band`**

moves within this amount are ignored UNITS ARE 1/16 COUNT

Definition at line 130 of file pgpmac.h.

**6.24.2.13 `Isredis_obj_t* lspmac_motor_struct::inactive_init`**

pmac commands to inactivate the motor

Definition at line 129 of file pgpmac.h.

**6.24.2.14 `int(* lspmac_motor_struct::jogAbs)(struct lspmac_motor_struct *, double)`**

function to move the motor

Definition at line 151 of file pgpmac.h.

**6.24.2.15 `double* lspmac_motor_struct::lut`**

lookup table (instead of u2c)

Definition at line 152 of file pgpmac.h.

**6.24.2.16 `int lspmac_motor_struct::magic`**

magic number identifying this as a motor structure

Definition at line 102 of file pgpmac.h.

**6.24.2.17 `Isredis_obj_t* lspmac_motor_struct::max_accel`**

our maximum acceleration (cts/msec<sup>^2</sup>)

Definition at line 131 of file pgpmac.h.

**6.24.2.18 `Isredis_obj_t* lspmac_motor_struct::max_pos`**

our maximum position (soft limit)

Definition at line 132 of file pgpmac.h.

**6.24.2.19 `Isredis_obj_t* lspmac_motor_struct::max_speed`**

our maximum speed (cts/msec)

Definition at line 133 of file pgpmac.h.

**6.24.2.20 `Isredis_obj_t* lspmac_motor_struct::min_pos`**

our minimum position (soft limit)

Definition at line 134 of file pgpmac.h.

**6.24.2.21 int lspmac\_motor\_struct::motion\_seen**

set to 1 when motion has been verified to have started

Definition at line 108 of file pgpmac.h.

**6.24.2.22 Isredis\_obj\_t\* lspmac\_motor\_struct::motor\_num**

pmac motor number

Definition at line 135 of file pgpmac.h.

**6.24.2.23 int(\* lspmac\_motor\_struct::moveAbs)(struct lspmac\_motor\_struct \*, double)**

function to move the motor

Definition at line 150 of file pgpmac.h.

**6.24.2.24 pthread\_mutex\_t lspmac\_motor\_struct::mutex**

coordinate waiting for motor to be done

Definition at line 103 of file pgpmac.h.

**6.24.2.25 char\* lspmac\_motor\_struct::name**

Name of motor as referred by ls database kvs table.

Definition at line 123 of file pgpmac.h.

**6.24.2.26 Isredis\_obj\_t\* lspmac\_motor\_struct::neg\_limit\_hit**

negative limit status

Definition at line 138 of file pgpmac.h.

**6.24.2.27 Isredis\_obj\_t\* lspmac\_motor\_struct::neutral\_pos**

zero offset

Definition at line 136 of file pgpmac.h.

**6.24.2.28 int lspmac\_motor\_struct::nlut**

length of lut

Definition at line 153 of file pgpmac.h.

**6.24.2.29 int lspmac\_motor\_struct::not\_done**

set to 1 when request is queued, zero after motion has toggled

Definition at line 105 of file pgpmac.h.

---

**6.24.2.30 `Isredis_obj_t* lspmac_motor_struct::pos_limit_hit`**

positive limit status

Definition at line 137 of file pgpmac.h.

**6.24.2.31 `double lspmac_motor_struct::position`**

scaled position

Definition at line 114 of file pgpmac.h.

**6.24.2.32 `pmac_cmd_queue_t* lspmac_motor_struct::pq`**

the queue item requesting motion. Used to check time request was made

Definition at line 109 of file pgpmac.h.

**6.24.2.33 `Isredis_obj_t* lspmac_motor_struct::precision`**

moves of less than this amount may be ignored

Definition at line 139 of file pgpmac.h.

**6.24.2.34 `Isredis_obj_t* lspmac_motor_struct::printf_fmt`**

printf format

Definition at line 140 of file pgpmac.h.

**6.24.2.35 `void(* lspmac_motor_struct::read)(struct lspmac_motor_struct *)`**

method to read the motor status and position

Definition at line 106 of file pgpmac.h.

**6.24.2.36 `int lspmac_motor_struct::read_mask`**

With read\_ptr find bit to read for binary i/o.

Definition at line 149 of file pgpmac.h.

**6.24.2.37 `int* lspmac_motor_struct::read_ptr`**

With read\_mask finds bit to read for binary i/o.

Definition at line 148 of file pgpmac.h.

**6.24.2.38 `Isredis_obj_t* lspmac_motor_struct::redis_fmt`**

special format string to create text array for putting the position back into redis

Definition at line 141 of file pgpmac.h.

**6.24.2.39 Isredis\_obj\_t\* Ispmac\_motor\_struct::redis\_position**

how we report our position to the world

Definition at line 142 of file pgpmac.h.

**6.24.2.40 double Ispmac\_motor\_struct::reported\_pg\_position**

previous position reported to postgresql

Definition at line 115 of file pgpmac.h.

**6.24.2.41 double Ispmac\_motor\_struct::reported\_position**

previous position reported to redis

Definition at line 116 of file pgpmac.h.

**6.24.2.42 int Ispmac\_motor\_struct::requested\_pos\_cnts**

requested position

Definition at line 111 of file pgpmac.h.

**6.24.2.43 double Ispmac\_motor\_struct::requested\_position**

The position as requested by the user.

Definition at line 117 of file pgpmac.h.

**6.24.2.44 int Ispmac\_motor\_struct::status1**

local copy of status1

Definition at line 119 of file pgpmac.h.

**6.24.2.45 int\* Ispmac\_motor\_struct::status1\_p**

First 24 bit PMAC motor status word.

Definition at line 118 of file pgpmac.h.

**6.24.2.46 int Ispmac\_motor\_struct::status2**

local copy of status2

Definition at line 121 of file pgpmac.h.

**6.24.2.47 int\* Ispmac\_motor\_struct::status2\_p**

Sectond 24 bit PMAC motor status word.

Definition at line 120 of file pgpmac.h.

---

#### 6.24.2.48 `Isredis_obj_t* lspmac_motor_struct::status_str`

A talky version of the status.

Definition at line 143 of file pgpmac.h.

#### 6.24.2.49 `Isredis_obj_t* lspmac_motor_struct::u2c`

conversion from counts to units: 0.0 means not loaded yet

Definition at line 144 of file pgpmac.h.

#### 6.24.2.50 `Isredis_obj_t* lspmac_motor_struct::unit`

string to use as the units

Definition at line 145 of file pgpmac.h.

#### 6.24.2.51 `Isredis_obj_t* lspmac_motor_struct::update_resolution`

Change needs to be at least this big to report as a new position to the database.

Definition at line 146 of file pgpmac.h.

#### 6.24.2.52 `WINDOW* lspmac_motor_struct::win`

our ncurses window

Definition at line 154 of file pgpmac.h.

#### 6.24.2.53 `char* lspmac_motor_struct::write_fmt`

Format string to write requested position to PMAC used for binary io.

Definition at line 147 of file pgpmac.h.

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

- [pgpmac.h](#)

## 6.25 `Isredis_obj_struct` Struct Reference

Redis Object Basic object whose value is synchronized with our redis db.

```
#include <pgpmac.h>
```

### Data Fields

- `pthread_mutex_t mutex`  
*Don't let anyone use an old value.*
- `pthread_cond_t cond`  
*wait for a valid value*
- struct `Isredis_obj_struct * next`  
*the next in our list (I guess this is going to be a linked list)*
- char `val`

- int **wait\_for\_me**  
*1 if we think the value is good, 0 otherwise*
- char \* **key**  
*The redis key for this object.*
- char \* **events\_name**  
*Name used to generate events (normally key without the station id)*
- int **value\_length**  
*Number of bytes allocated for value (not value's string length)*
- char \* **value**  
*our value*
- double **dvalue**  
*our value as a double*
- long int **lvalue**  
*our value as a long*
- char \*\* **avalue**  
*our value as an array of strings*
- int **bvalue**  
*our value as a boolean (1 or 0) -1 means we couldn't figure it out*
- char **cvalue**  
*just the first character of our value*
- int **hits**  
*number of times we've searched for this key*

### 6.25.1 Detailed Description

Redis Object Basic object whose value is synchronized with our redis db.

Definition at line 38 of file pgpmac.h.

### 6.25.2 Field Documentation

#### 6.25.2.1 char\*\* lsredis\_obj\_struct::avalue

our value as an array of strings

Definition at line 50 of file pgpmac.h.

#### 6.25.2.2 int lsredis\_obj\_struct::bvalue

our value as a boolean (1 or 0) -1 means we couldn't figure it out

Definition at line 51 of file pgpmac.h.

#### 6.25.2.3 pthread\_cond\_t lsredis\_obj\_struct::cond

wait for a valid value

Definition at line 40 of file pgpmac.h.

#### 6.25.2.4 char lsredis\_obj\_struct::cvalue

just the first character of our value

Definition at line 52 of file pgpmac.h.

**6.25.2.5 double lsredis\_obj\_struct::dvalue**

our value as a double

Definition at line 48 of file pgpmac.h.

**6.25.2.6 char\* lsredis\_obj\_struct::events\_name**

Name used to generate events (normally key without the station id)

Definition at line 45 of file pgpmac.h.

**6.25.2.7 int lsredis\_obj\_struct::hits**

number of times we've searched for this key

Definition at line 53 of file pgpmac.h.

**6.25.2.8 char\* lsredis\_obj\_struct::key**

The redis key for this object.

Definition at line 44 of file pgpmac.h.

**6.25.2.9 long int lsredis\_obj\_struct::lvalue**

our value as a long

Definition at line 49 of file pgpmac.h.

**6.25.2.10 pthread\_mutex\_t lsredis\_obj\_struct::mutex**

Don't let anyone use an old value.

Definition at line 39 of file pgpmac.h.

**6.25.2.11 struct lsredis\_obj\_struct\* lsredis\_obj\_struct::next**

the next in our list (I guess this is going to be a linked list)

Definition at line 41 of file pgpmac.h.

**6.25.2.12 char lsredis\_obj\_struct::valid**

1 if we think the value is good, 0 otherwise

Definition at line 42 of file pgpmac.h.

**6.25.2.13 char\* lsredis\_obj\_struct::value**

our value

Definition at line 47 of file pgpmac.h.

**6.25.2.14 int lsredis\_obj\_struct::value\_length**

Number of bytes allocated for value (not value's string length)

Definition at line 46 of file pgpmac.h.

**6.25.2.15 int lsredis\_obj\_struct::wait\_for\_me**

Number of times we need to see our publication before we start accepting new values.

Definition at line 43 of file pgpmac.h.

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

- [pgpmac.h](#)

## 6.26 lsredis\_preset\_list\_struct Struct Reference

### Data Fields

- struct [lsredis\\_preset\\_list\\_struct](#) \* `next`
- char \* `key`
- int `index`
- [lsredis\\_obj\\_t](#) \* `name`
- [lsredis\\_obj\\_t](#) \* `position`

### 6.26.1 Detailed Description

Definition at line 97 of file lsredis.c.

### 6.26.2 Field Documentation

**6.26.2.1 int lsredis\_preset\_list\_struct::index**

Definition at line 100 of file lsredis.c.

**6.26.2.2 char\* lsredis\_preset\_list\_struct::key**

Definition at line 99 of file lsredis.c.

**6.26.2.3 lsredis\_obj\_t\* lsredis\_preset\_list\_struct::name**

Definition at line 101 of file lsredis.c.

**6.26.2.4 struct lsredis\_preset\_list\_struct\* lsredis\_preset\_list\_struct::next**

Definition at line 98 of file lsredis.c.

### 6.26.2.5 `lsredis_obj_t* lsredis_preset_list_struct::position`

Definition at line 102 of file `lsredis.c`.

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

- [lsredis.c](#)

## 6.27 `Istimer_list_struct` Struct Reference

Everything we need to know about a timer.

### Data Fields

- int `shots`  
*run this many times: -1 means reload forever, 0 means we are done with this timer and it may be reused*
- unsigned long int `ncalls`  
*track how many times we triggered a callback (like an unsigned long int is really needed)*
- char `event [LSEVENTS_EVENT_LENGTH]`  
*the event to send*
- long int `next_secs`  
*epoch (seconds) of next alarm*
- long int `next_nsecs`  
*nano seconds of next alarm*
- long int `delay_secs`  
*number of seconds for a periodic delay*
- long int `delay_nsecs`  
*nano seconds of delay*
- long int `last_secs`  
*the last time this timer was triggered*
- long int `last_nsecs`  
*the last time this timer was triggered*
- long int `init_secs`  
*our initialization time*
- long int `init_nsecs`  
*our initialization time*

### 6.27.1 Detailed Description

Everything we need to know about a timer.

Definition at line 22 of file `Istimer.c`.

### 6.27.2 Field Documentation

#### 6.27.2.1 long int `Istimer_list_struct::delay_nsecs`

nano seconds of delay

Definition at line 29 of file `Istimer.c`.

**6.27.2.2 long int Istimer\_list\_struct::delay\_secs**

number of seconds for a periodic delay

Definition at line 28 of file Istimer.c.

**6.27.2.3 char Istimer\_list\_struct::event[LSEVENTS\_EVENT\_LENGTH]**

the event to send

Definition at line 25 of file Istimer.c.

**6.27.2.4 long int Istimer\_list\_struct::init\_nsecs**

our initialization time

Definition at line 33 of file Istimer.c.

**6.27.2.5 long int Istimer\_list\_struct::init\_secs**

our initialization time

Definition at line 32 of file Istimer.c.

**6.27.2.6 long int Istimer\_list\_struct::last\_nsecs**

the last time this timer was triggered

Definition at line 31 of file Istimer.c.

**6.27.2.7 long int Istimer\_list\_struct::last\_secs**

the last time this timer was triggered

Definition at line 30 of file Istimer.c.

**6.27.2.8 unsigned long int Istimer\_list\_struct::ncalls**

track how many times we triggered a callback (like an unsigned long int is really needed)

Definition at line 24 of file Istimer.c.

**6.27.2.9 long int Istimer\_list\_struct::next\_nsecs**

nano seconds of next alarm

Definition at line 27 of file Istimer.c.

**6.27.2.10 long int Istimer\_list\_struct::next\_secs**

epoch (seconds) of next alarm

Definition at line 26 of file Istimer.c.

### 6.27.2.11 int lstimber\_list\_struct::shots

run this many times: -1 means reload forever, 0 means we are done with this timer and it may be reused

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

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

- [lstimber.c](#)

## 6.28 md2cmds\_cmd\_kv\_struct Struct Reference

### Data Fields

- char \* [k](#)
- int(\* [v](#))(const char \*)

### 6.28.1 Detailed Description

Definition at line 39 of file [md2cmds.c](#).

### 6.28.2 Field Documentation

#### 6.28.2.1 char\* md2cmds\_cmd\_kv\_struct::k

Definition at line 40 of file [md2cmds.c](#).

#### 6.28.2.2 int(\* md2cmds\_cmd\_kv\_struct::v)(const char \*)

Definition at line 41 of file [md2cmds.c](#).

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

- [md2cmds.c](#)

## 6.29 md2StatusStruct Struct Reference

The block of memory retrieved in a status request.

### Data Fields

- int [dummy1](#)
- int [omega\\_status\\_1](#)
- int [alignx\\_status\\_1](#)
- int [aligny\\_status\\_1](#)
- int [alignz\\_status\\_1](#)
- int [analyzer\\_status\\_1](#)
- int [zoom\\_status\\_1](#)
- int [aperturey\\_status\\_1](#)
- int [aperturez\\_status\\_1](#)
- int [capy\\_status\\_1](#)
- int [capz\\_status\\_1](#)

- int `scint_status_1`
- int `centerx_status_1`
- int `centery_status_1`
- int `kappa_status_1`
- int `phi_status_1`
- int `dummy2`
- int `omega_status_2`
- int `alignx_status_2`
- int `aligny_status_2`
- int `alignz_status_2`
- int `analyzer_status_2`
- int `zoom_status_2`
- int `aperturey_status_2`
- int `aperturez_status_2`
- int `capy_status_2`
- int `capz_status_2`
- int `scint_status_2`
- int `centerx_status_2`
- int `centery_status_2`
- int `kappa_status_2`
- int `phi_status_2`
- int `dummy3`
- int `omega_act_pos`
- int `alignx_act_pos`
- int `aligny_act_pos`
- int `alignz_act_pos`
- int `analyzer_act_pos`
- int `zoom_act_pos`
- int `aperturey_act_pos`
- int `aperturez_act_pos`
- int `capy_act_pos`
- int `capz_act_pos`
- int `scint_act_pos`
- int `centerx_act_pos`
- int `centery_act_pos`
- int `kappa_act_pos`
- int `phi_act_pos`
- int `acc11c_1`
- int `acc11c_2`
- int `acc11c_3`
- int `acc11c_5`
- int `acc11c_6`
- int `front_dac`
- int `back_dac`
- int `scint_piezo`
- int `dummy4`
- int `dummy5`
- int `dummy6`
- int `dummy7`
- int `dummy8`
- int `dummy9`
- int `dummyA`
- int `dummyB`
- int `fs_is_open`
- int `phiscan`

- int `fs_has_opened`
- int `fs_has_opened_globally`
- int `number_passes`
- int `moving_flags`

### 6.29.1 Detailed Description

The block of memory retrieved in a status request.

Definition at line 258 of file `Ispmac.c`.

### 6.29.2 Field Documentation

#### 6.29.2.1 int md2StatusStruct::acc11c\_1

Definition at line 325 of file `Ispmac.c`.

#### 6.29.2.2 int md2StatusStruct::acc11c\_2

Definition at line 326 of file `Ispmac.c`.

#### 6.29.2.3 int md2StatusStruct::acc11c\_3

Definition at line 327 of file `Ispmac.c`.

#### 6.29.2.4 int md2StatusStruct::acc11c\_5

Definition at line 328 of file `Ispmac.c`.

#### 6.29.2.5 int md2StatusStruct::acc11c\_6

Definition at line 329 of file `Ispmac.c`.

#### 6.29.2.6 int md2StatusStruct::alignx\_act\_pos

Definition at line 309 of file `Ispmac.c`.

#### 6.29.2.7 int md2StatusStruct::alignx\_status\_1

Definition at line 275 of file `Ispmac.c`.

#### 6.29.2.8 int md2StatusStruct::alignx\_status\_2

Definition at line 292 of file `Ispmac.c`.

#### 6.29.2.9 int md2StatusStruct::aligny\_act\_pos

Definition at line 310 of file `Ispmac.c`.

6.29.2.10 int md2StatusStruct::aligny\_status\_1

Definition at line 276 of file lspmac.c.

6.29.2.11 int md2StatusStruct::aligny\_status\_2

Definition at line 293 of file lspmac.c.

6.29.2.12 int md2StatusStruct::alignz\_act\_pos

Definition at line 311 of file lspmac.c.

6.29.2.13 int md2StatusStruct::alignz\_status\_1

Definition at line 277 of file lspmac.c.

6.29.2.14 int md2StatusStruct::alignz\_status\_2

Definition at line 294 of file lspmac.c.

6.29.2.15 int md2StatusStruct::analyzer\_act\_pos

Definition at line 312 of file lspmac.c.

6.29.2.16 int md2StatusStruct::analyzer\_status\_1

Definition at line 278 of file lspmac.c.

6.29.2.17 int md2StatusStruct::analyzer\_status\_2

Definition at line 295 of file lspmac.c.

6.29.2.18 int md2StatusStruct::aperturey\_act\_pos

Definition at line 314 of file lspmac.c.

6.29.2.19 int md2StatusStruct::aperturey\_status\_1

Definition at line 280 of file lspmac.c.

6.29.2.20 int md2StatusStruct::aperturey\_status\_2

Definition at line 297 of file lspmac.c.

6.29.2.21 int md2StatusStruct::aperturez\_act\_pos

Definition at line 315 of file lspmac.c.

6.29.2.22 int md2StatusStruct::aperturez\_status\_1

Definition at line 281 of file lspmac.c.

6.29.2.23 int md2StatusStruct::aperturez\_status\_2

Definition at line 298 of file lspmac.c.

6.29.2.24 int md2StatusStruct::back\_dac

Definition at line 331 of file lspmac.c.

6.29.2.25 int md2StatusStruct::capy\_act\_pos

Definition at line 316 of file lspmac.c.

6.29.2.26 int md2StatusStruct::capy\_status\_1

Definition at line 282 of file lspmac.c.

6.29.2.27 int md2StatusStruct::capy\_status\_2

Definition at line 299 of file lspmac.c.

6.29.2.28 int md2StatusStruct::capz\_act\_pos

Definition at line 317 of file lspmac.c.

6.29.2.29 int md2StatusStruct::capz\_status\_1

Definition at line 283 of file lspmac.c.

6.29.2.30 int md2StatusStruct::capz\_status\_2

Definition at line 300 of file lspmac.c.

6.29.2.31 int md2StatusStruct::centerx\_act\_pos

Definition at line 319 of file lspmac.c.

6.29.2.32 int md2StatusStruct::centerx\_status\_1

Definition at line 285 of file lspmac.c.

6.29.2.33 int md2StatusStruct::centerx\_status\_2

Definition at line 302 of file lspmac.c.

6.29.2.34 int md2StatusStruct::centery\_act\_pos

Definition at line 320 of file lspmac.c.

6.29.2.35 int md2StatusStruct::centery\_status\_1

Definition at line 286 of file lspmac.c.

6.29.2.36 int md2StatusStruct::centery\_status\_2

Definition at line 303 of file lspmac.c.

6.29.2.37 int md2StatusStruct::dummy1

Definition at line 273 of file lspmac.c.

6.29.2.38 int md2StatusStruct::dummy2

Definition at line 290 of file lspmac.c.

6.29.2.39 int md2StatusStruct::dummy3

Definition at line 307 of file lspmac.c.

6.29.2.40 int md2StatusStruct::dummy4

Definition at line 334 of file lspmac.c.

6.29.2.41 int md2StatusStruct::dummy5

Definition at line 335 of file lspmac.c.

6.29.2.42 int md2StatusStruct::dummy6

Definition at line 336 of file lspmac.c.

6.29.2.43 int md2StatusStruct::dummy7

Definition at line 337 of file lspmac.c.

6.29.2.44 int md2StatusStruct::dummy8

Definition at line 338 of file lspmac.c.

6.29.2.45 int md2StatusStruct::dummy9

Definition at line 339 of file lspmac.c.

6.29.2.46 int md2StatusStruct::dummyA

Definition at line 340 of file lspmac.c.

6.29.2.47 int md2StatusStruct::dummyB

Definition at line 341 of file lspmac.c.

6.29.2.48 int md2StatusStruct::front\_dac

Definition at line 330 of file lspmac.c.

6.29.2.49 int md2StatusStruct::fs\_has\_opened

Definition at line 345 of file lspmac.c.

6.29.2.50 int md2StatusStruct::fs\_has\_opened\_globally

Definition at line 346 of file lspmac.c.

6.29.2.51 int md2StatusStruct::fs\_is\_open

Definition at line 343 of file lspmac.c.

6.29.2.52 int md2StatusStruct::kappa\_act\_pos

Definition at line 321 of file lspmac.c.

6.29.2.53 int md2StatusStruct::kappa\_status\_1

Definition at line 287 of file lspmac.c.

6.29.2.54 int md2StatusStruct::kappa\_status\_2

Definition at line 304 of file lspmac.c.

6.29.2.55 int md2StatusStruct::moving\_flags

Definition at line 349 of file lspmac.c.

6.29.2.56 int md2StatusStruct::number\_passes

Definition at line 347 of file lspmac.c.

6.29.2.57 int md2StatusStruct::omega\_act\_pos

Definition at line 308 of file lspmac.c.

6.29.2.58 int md2StatusStruct::omega\_status\_1

Definition at line 274 of file lspmac.c.

6.29.2.59 int md2StatusStruct::omega\_status\_2

Definition at line 291 of file lspmac.c.

6.29.2.60 int md2StatusStruct::phi\_act\_pos

Definition at line 322 of file lspmac.c.

6.29.2.61 int md2StatusStruct::phi\_status\_1

Definition at line 288 of file lspmac.c.

6.29.2.62 int md2StatusStruct::phi\_status\_2

Definition at line 305 of file lspmac.c.

6.29.2.63 int md2StatusStruct::phiscan

Definition at line 344 of file lspmac.c.

6.29.2.64 int md2StatusStruct::scint\_act\_pos

Definition at line 318 of file lspmac.c.

6.29.2.65 int md2StatusStruct::scint\_piezo

Definition at line 332 of file lspmac.c.

6.29.2.66 int md2StatusStruct::scint\_status\_1

Definition at line 284 of file lspmac.c.

6.29.2.67 int md2StatusStruct::scint\_status\_2

Definition at line 301 of file lspmac.c.

6.29.2.68 int md2StatusStruct::zoom\_act\_pos

Definition at line 313 of file lspmac.c.

6.29.2.69 int md2StatusStruct::zoom\_status\_1

Definition at line 279 of file lspmac.c.

### 6.29.2.70 int md2StatusStruct::zoom\_status\_2

Definition at line 296 of file lspmac.c.

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

- [lspmac.c](#)

## 6.30 tagEthernetCmd Struct Reference

PMAC ethernet packet definition.

```
#include <pgpmac.h>
```

### Data Fields

- unsigned char [RequestType](#)  
*VR\_UPLOAD or VR\_DOWNLOAD.*
- unsigned char [Request](#)  
*The command to run (VR\_PMAC\_GETMEM, etc).*
- unsigned short [wValue](#)  
*Command parameter 1.*
- unsigned short [wIndex](#)  
*Command parameter 2.*
- unsigned short [wLength](#)  
*Number of bytes in bData.*
- unsigned char [bData](#) [1492]  
*The data buffer, if required.*

### 6.30.1 Detailed Description

PMAC ethernet packet definition.

Taken directly from the Delta Tau documentation.

Definition at line 73 of file pgpmac.h.

### 6.30.2 Field Documentation

#### 6.30.2.1 unsigned char tagEthernetCmd::bData[1492]

The data buffer, if required.

Definition at line 79 of file pgpmac.h.

#### 6.30.2.2 unsigned char tagEthernetCmd::Request

The command to run (VR\_PMAC\_GETMEM, etc).

Definition at line 75 of file pgpmac.h.

**6.30.2.3 unsigned char tagEthernetCmd::RequestType**

VR\_UPLOAD or VR\_DOWNLOAD.

Definition at line 74 of file pgpmac.h.

**6.30.2.4 unsigned short tagEthernetCmd::wIndex**

Command parameter 2.

Definition at line 77 of file pgpmac.h.

**6.30.2.5 unsigned short tagEthernetCmd::wLength**

Number of bytes in bData.

Definition at line 78 of file pgpmac.h.

**6.30.2.6 unsigned short tagEthernetCmd::wValue**

Command parameter 1.

Definition at line 76 of file pgpmac.h.

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

- [pgpmac.h](#)



# Chapter 7

## File Documentation

### 7.1 iniParser.py File Reference

#### Data Structures

- class `iniParser.iniParser`

*This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.*

#### Namespaces

- namespace `iniParser`

#### Variables

- tuple `iniParser.ip` `iniParser( "21-ID-D/microdiff_pref.ini")`

### 7.2 lsevents.c File Reference

event subsystem for inter-pgpmac communication

```
#include "pgpmac.h"
```

#### Data Structures

- struct `lsevents_queue_struct`

*Storage definition for the events.*
- struct `lsevents_listener_struct`

*Linked list of event listeners.*
- struct `lsevents_callbacks_struct`

*lsevents linked list of callbacks for each event*
- struct `lsevents_event_names_struct`

*linked list of all the event names used to regenerate the hash table*

## Macros

- #define LSEVENTS\_QUEUE\_LENGTH 512

## Typedefs

- typedef struct  
`Isevents_queue_struct` `Isevents_queue_t`  
*Storage definition for the events.*
- typedef struct  
`Isevents_listener_struct` `Isevents_listener_t`  
*Linked list of event listeners.*
- typedef struct  
`Isevents_callbacks_struct` `Isevents_callbacks_t`  
*Isevents linked list of callbacks for each event*
- typedef struct  
`Isevents_event_names_struct` `Isevents_event_names_t`  
*linked list of all the event names used to regenerate the hash table*

## Functions

- void `Isevents_send_event` (char \*fmt,...)  
*Call the callback routines for the given event.*
- void `Isevents_add_listener` (char \*raw\_regexp, void(\*cb)(char \*))  
*Add a callback routine to listen for a specific event.*
- void `Isevents_remove_listener` (char \*event, void(\*cb)(char \*))  
*Remove a listener previously added with `Isevents_add_listener`.*
- `Isevents_callbacks_t *` `Isevents_register_event` (char \*event)  
*Add a new event name and find matching callbacks as a returned linked list.*
- void `Isevents_preregister_event` (char \*fmt,...)
- void \* `Isevents_worker` (void \*dummy)  
*Our worker.*
- void `Isevents_init` ()  
*Initialize this module.*
- void `Isevents_run` ()  
*Start up the thread and get out of the way.*

## Variables

- static `Isevents_queue_t` `Isevents_queue` [LSEVENTS\_QUEUE\_LENGTH]  
*simple list of events*
- static unsigned int `Isevents_queue_on` = 0  
*next queue location to write*
- static unsigned int `Isevents_queue_off` = 0  
*next queue location to read*
- static int `Isevents_max_events` = 1024
- static int `Isevents_n_events` = 0
- static struct hsearch\_data `Isevents_event_name_ht`
- static `Isevents_listener_t *` `Isevents_listeners_p` = NULL  
*Pointer to the first item in the link list of listeners.*
- static `Isevents_event_names_t *` `Isevents_event_names` = NULL

- static pthread\_t [lsevents\\_thread](#)  
*thread to run the event queue*
- static pthread\_mutex\_t [lsevents\\_listener\\_mutex](#)  
*mutex to protect the listener linked list*
- static pthread\_mutex\_t [lsevents\\_queue\\_mutex](#)  
*mutex to protect the event queue*
- static pthread\_cond\_t [lsevents\\_queue\\_cond](#)  
*condition to pause the queue if needed*

### 7.2.1 Detailed Description

event subsystem for inter-pgpmac communication

#### Date

2012

#### Author

Keith Brister

#### Copyright

All Rights Reserved

Definition in file [lsevents.c](#).

### 7.2.2 Macro Definition Documentation

#### 7.2.2.1 #define LSEVENTS\_QUEUE\_LENGTH 512

Definition at line 10 of file lsevents.c.

### 7.2.3 Typedef Documentation

#### 7.2.3.1 [typedef struct lsevents\\_callbacks\\_struct lsevents\\_callbacks\\_t](#)

lsevents linked list of callbacks for each event

#### 7.2.3.2 [typedef struct lsevents\\_event\\_names\\_struct lsevents\\_event\\_names\\_t](#)

linked list of all the event names used to regenerate the hash table

#### 7.2.3.3 [typedef struct lsevents\\_listener\\_struct lsevents\\_listener\\_t](#)

Linked list of event listeners.

#### 7.2.3.4 [typedef struct lsevents\\_queue\\_struct lsevents\\_queue\\_t](#)

Storage definition for the events.

Just a string for now. Perhaps one day we'll succumb to the temptation to add an argument or two.

## 7.2.4 Function Documentation

### 7.2.4.1 void lsevents\_add\_listener ( char \* raw\_regexp, void(\*)(char \*) cb )

Add a callback routine to listen for a specific event.

#### Parameters

<i>raw_regexp</i>	String value of regular expression to listen to
<i>cb</i>	the routine to call

Definition at line 99 of file lsevents.c.

```

    {
lsevents_listener_t      *new;
lsevents_event_names_t   *enp;
lsevents_callbacks_t     *cbp;
int err;
char *errbuf;
int nerrbuf;

new = calloc( 1, sizeof( lsevents_listener_t));
if( new == NULL) {
    lslogging_log_message( "lsevents_add_listener: out of
        memory");
    exit( -1);
}

err = regcomp( &new->re, raw_regexp, REG_EXTENDED | REG_NOSUB);
if( err != 0) {
    nerrbuf = regerror( err, &new->re, NULL, 0);
    errbuf = calloc( nerrbuf, sizeof( char));
    if( errbuf == NULL) {
        lslogging_log_message( "lsevents_add_listener: out
            of memory (re)");
        exit( -1);
    }
    regerror( err, &new->re, errbuf, nerrbuf);
//    lslogging_log_message( "lsevents_add_listener: %s", errbuf);
    free( errbuf);
    free( new);
    return;
}

new->raw_regexp = strdup( raw_regexp);
new->cb      = cb;

pthread_mutex_lock( &lsevents_listener_mutex);
new->next = lsevents_listeners_p;
lsevents_listeners_p = new;

for( enp = lsevents_event_names; enp != NULL; enp = enp->
    next) {
    if( regexec( &new->re, enp->event, 0, NULL, 0) == 0) {
        cbp      = calloc( 1, sizeof( lsevents_callbacks_t))
        ;
        cbp->cb      = cb;
        cbp->next = enp->cb;
        enp->cb = cbp;
    }
}

pthread_mutex_unlock( &lsevents_listener_mutex);

// lslogging_log_message( "lsevents_add_listener: added listener for event
// '%s'", raw_regexp);
}

```

### 7.2.4.2 void lsevents\_init ( )

Initialize this module.

Definition at line 373 of file lsevents.c.

```

    {
pthread_mutexattr_t mutex_initializer;

// Use recursive mutexes
//
pthread_mutexattr_init( &mutex_initializer);
pthread_mutexattr_settype( &mutex_initializer, PTHREAD_MUTEX_RECURSIVE);

pthread_mutex_init( &lsevents_queue_mutex,   &
                   mutex_initializer);
pthread_cond_init( &lsevents_queue_cond,      NULL);
pthread_mutex_init( &lsevents_listener_mutex, &
                   mutex_initializer);

hcreate_r( 2*lsevents_max_events, &lsevents_event_name_ht
        );
}

```

#### 7.2.4.3 void lsevents\_preregister\_event( char \* fmt, ... )

Definition at line 314 of file lsevents.c.

```

{
char s[128];
va_list arg_ptr;

va_start( arg_ptr, fmt);
vsnprintf( s, sizeof( s) - 1, fmt, arg_ptr);
s[sizeof(s)-1] = 0;
va_end( arg_ptr);

lsevents_register_event( s);
}

```

#### 7.2.4.4 lsevents\_callbacks\_t\* lsevents\_register\_event( char \* event )

Add a new event name and find matching callbacks as a returned linked list.

Definition at line 221 of file lsevents.c.

```

{
ENTRY entry_in, *entry_outp;
int err;
lsevents_callbacks_t *new_cb;
lsevents_event_names_t *new_event_name, *enp;
lsevents_listener_t *p;

//
// Search for event
//
entry_in.key = event;
entry_in.data = NULL;

pthread_mutex_lock( &lsevents_listener_mutex);
err = hsearch_r( entry_in, FIND, &entry_outp, &lsevents_event_name_ht
                 );
if( err != 0) {
    //
    // Success, we found the entry
    //
    enp = entry_outp->data;
    pthread_mutex_unlock( &lsevents_listener_mutex);
    return enp->cbl;
}

if( errno != ESRCH) {
    //
    // Something awful happened. At least log it
    //
    lslogging_log_message( "lsevents_register_event:
                           hsearch_r returnd %d: %s", errno, strerror( errno));
    pthread_mutex_unlock( &lsevents_listener_mutex);
    return NULL;
}

// lslogging_log_message( "lsevents_register_event: adding event '%s'",

```

```

        event);
// Not Found
//
// Create new event name item
new_event_name = calloc( 1, sizeof( lsevents_event_names_t
));
new_event_name->event = strdup( event);
new_event_name->cbl = NULL;

//
// Find matching callbacks
//
for( p = lsevents_listeners_p; p != NULL; p = p->next
) {
    if( regexec( &p->re, event, 0, NULL, 0 ) == 0) {
        new_cb = calloc( 1, sizeof( lsevents_callbacks_t));
        new_cb->cb = p->cb;
        new_cb->next = new_event_name->cbl;
        new_event_name->cbl = new_cb;
    }
}

//
// Add the new event to our linked list
//
new_event_name->next = lsevents_event_names;
lsevents_event_names = new_event_name;

//
// Also add the new event to our hash table
//
entry_in.key = new_event_name->event;
entry_in.data = new_event_name;
err = hsearch_r( entry_in, ENTER, &entry_outp, &lsevents_event_name_ht
);
if( err == 0) {
    //
    // Something bad happened but we can still return a valid callback list. We
    // just can't use the hash table to find it again later
    //
    lslogging_log_message( "lsevents_register_event: Could
        not add event name: hsearch_r returned %d: %s", errno, strerror( errno));
    pthread_mutex_unlock( &lsevents_listener_mutex);
    return new_event_name->cbl;
}

if( ++lsevents_n_events >= lsevents_max_events
) {
    hdestroy_r( &lsevents_event_name_ht);
    lslogging_log_message( "lsevents_register_event:
        Increasing event name hash table to %d. lsevents_n_events=%d", 2 *
    lsevents_max_events, lsevents_n_events);
    lsevents_max_events *= 2;
    hcreate_r( lsevents_max_events * 2, &
    lsevents_event_name_ht);
    for( enp = lsevents_event_names; enp != NULL; enp = enp
->next) {
        entry_in.key = enp->event;
        entry_in.data = enp;
        hsearch_r( entry_in, ENTER, &entry_outp, &lsevents_event_name_ht
);
    }
}
// lslogging_log_message( "lsevents_register_event: added event '%s'",
// event);
pthread_mutex_unlock( &lsevents_listener_mutex);
return new_event_name->cbl;
}

```

#### 7.2.4.5 void lsevents\_remove\_listener ( char \* event, void(\*)(char \*) cb )

Remove a listener previously added with lsevents\_add\_listener.

##### Parameters

<i>event</i>	The name of the event (possibly a regular expression string)
<i>cb</i>	The callback routine to remove

Definition at line 157 of file lsevents.c.

```

{
lsevents_listener_t *last, *current;
lsevents_event_names_t *enp;
lsevents_callbacks_t *cbp, *last_cbp;

//
// Find the listener to remove
// and unlink it from the list
//
pthread_mutex_lock( &lsevents_listener_mutex);
last = NULL;
for( current = lsevents_listeners_p; current != NULL;
    current = current->next) {
    if( strcmp( last->raw_regexp, event) == 0 && last->cb == cb) {
        if( last == NULL) {
            lsevents_listeners_p = current->next;
        } else {
            last->next = current->next;
        }
        break;
    }
    last = current;
}

if( current == NULL) {
    lslogging_log_message( "lsevents_remove_listener:
        Could not find this listener for event '%s'", event);
    pthread_mutex_unlock( &lsevents_listener_mutex);
    return;
}

//
// Remove callback from lists of event names
//
for( enp = lsevents_event_names; enp != NULL; enp = enp->
    next) {
    if( regexec( &current->re, enp->event, 0, NULL, 0) == 0) {
        last_cbp = NULL;
        for( cbp = enp->cbl; cbp != NULL; cbp = cbp->next) {
            if( cbp->cb == cb) {
                if( last_cbp == NULL)
                    enp->cbl = NULL;
                else
                    last_cbp->next = cbp->next;
                free( cbp);
                break;
            }
        }
    }
}

pthread_mutex_unlock( &lsevents_listener_mutex);

//
// Now remove it
//
if( current->raw_regexp != NULL)
    free( current->raw_regexp);
free( current);
}
}

```

#### 7.2.4.6 void lsevents\_run ( )

Start up the thread and get out of the way.

Definition at line 390 of file lsevents.c.

```

{
pthread_create( &lsevents_thread, NULL, lsevents_worker
    , NULL);
}

```

#### 7.2.4.7 void lsevents\_send\_event ( char \* fmt, ... )

Call the callback routines for the given event.

**Parameters**

<i>fmt</i>	a printf style formating string
...	list of arguments specified by the format string

Definition at line 73 of file lsevents.c.

```

{
char event[LSEVENTS_EVENT_LENGTH];
va_list arg_ptr;

va_start( arg_ptr, fmt);
vsnprintf( event, sizeof(event)-1, fmt, arg_ptr);
event[sizeof(event)-1]=0;
va_end( arg_ptr);

pthread_mutex_lock( &lsevents_queue_mutex);

// maybe wait for room on the queue
while( (lsevents_queue_on + 1) % LSEVENTS_QUEUE_LENGTH
      == lsevents_queue_off % LSEVENTS_QUEUE_LENGTH
      )
pthread_cond_wait( &lsevents_queue_cond, &
lsevents_queue_mutex);

lsevents_queue[(lsevents_queue_on++) % LSEVENTS_QUEUE_LENGTH].evp = strdup(event);

pthread_cond_signal( &lsevents_queue_cond);
pthread_mutex_unlock( &lsevents_queue_mutex);
}

```

#### 7.2.4.8 void\* lsevents\_worker ( void \* *dummy* )

Our worker.

**Parameters**

<i>dummy</i>	Unused but needed by pthreads to be happy
--------------	---

Definition at line 331 of file lsevents.c.

```

{
char *event;
lsevents_callbacks_t *cbi;

while( 1) {
pthread_mutex_lock( &lsevents_queue_mutex);

//
// wait for someone to send an event
//
while( lsevents_queue_off == lsevents_queue_on
      )
pthread_cond_wait( &lsevents_queue_cond, &
lsevents_queue_mutex);

//
// Get our event name
//
event = lsevents_queue[(lsevents_queue_off+
+) % LSEVENTS_QUEUE_LENGTH].evp;

//
// let the send event process know there is room on the queue again
//
pthread_cond_signal( &lsevents_queue_cond);
pthread_mutex_unlock( &lsevents_queue_mutex);

//
// call our callbacks
//
pthread_mutex_lock( &lsevents_listener_mutex);
for( cbi = lsevents_register_event( event); cbi !=
NULL; cbi = cbi->next) {
cbi->cb( event);
}
}
```

```
    pthread_mutex_unlock( &lsevents_listener_mutex);  
    free( event);  
}  
return NULL;  
}
```

## 7.2.5 Variable Documentation

### 7.2.5.1 struct hsearch\_data lsevents\_event\_name\_ht [static]

Definition at line 31 of file lsevents.c.

### 7.2.5.2 lsevents\_event\_names\_t\* lsevents\_event\_names = NULL [static]

Definition at line 60 of file lsevents.c.

### 7.2.5.3 pthread\_mutex\_t lsevents\_listener\_mutex [static]

mutex to protect the listener linked list

Definition at line 65 of file lsevents.c.

### 7.2.5.4 lsevents\_listener\_t\* lsevents\_listeners\_p = NULL [static]

Pointer to the first item in the link list of listeners.

Definition at line 42 of file lsevents.c.

### 7.2.5.5 int lsevents\_max\_events = 1024 [static]

Definition at line 29 of file lsevents.c.

### 7.2.5.6 int lsevents\_n\_events = 0 [static]

Definition at line 30 of file lsevents.c.

### 7.2.5.7 lsevents\_queue\_t lsevents\_queue[LSEVENTS\_QUEUE\_LENGTH] [static]

simple list of events

Definition at line 21 of file lsevents.c.

### 7.2.5.8 pthread\_cond\_t lsevents\_queue\_cond [static]

condition to pause the queue if needed

Definition at line 67 of file lsevents.c.

### 7.2.5.9 pthread\_mutex\_t lsevents\_queue\_mutex [static]

mutex to protect the event queue

Definition at line 66 of file lsevents.c.

### 7.2.5.10 `unsigned int lsevents.queue.off = 0 [static]`

next queue location to read

Definition at line 23 of file lsevents.c.

### 7.2.5.11 `unsigned int lsevents.queue.on = 0 [static]`

next queue location to write

Definition at line 22 of file lsevents.c.

### 7.2.5.12 `pthread_t lsevents.thread [static]`

thread to run the event queue

Definition at line 64 of file lsevents.c.

## 7.3 lslogging.c File Reference

Logs messages to a file.

```
#include "pgpmac.h"
```

### Data Structures

- struct [lslogging\\_queue\\_struct](#)

*Our log object: time and message.*

### Macros

- #define [LSLOGGING\\_FILE\\_NAME](#) "/tmp/pgpmac.log"

*Full name of the log file.*

- #define [LSLOGGING\\_MSG\\_LENGTH](#) 2048

*Fixed maximum length messages to keep some form of sanity.*

- #define [LSLOGGING\\_QUEUE\\_LENGTH](#) 8192

*Modest length queue.*

### Typedefs

- typedef struct [lslogging\\_queue\\_struct](#) [lslogging\\_queue\\_t](#)

*Our log object: time and message.*

### Functions

- void [lslogging\\_init\(\)](#)

*Initialize the lslogging objects.*

- void [lslogging\\_log\\_message\(char \\*fmt,...\)](#)

*The routine everyone will be talking about.*

- void [lslogging\\_event\\_cb\(char \\*event\)](#)

- *Log most events.*
- void \* **lslogging\_worker** (void \*dummy)
 

*Service the queue, write to the file.*
- void **lslogging\_run** ()
 

*Start up the worker thread.*

## Variables

- static pthread\_t **lslogging\_thread**

*our thread*
- static pthread\_mutex\_t **lslogging\_mutex**

*mutex to keep the various threads from adding to the queue at the exact same time*
- static pthread\_cond\_t **lslogging\_cond**

*We'll spend most of our time waiting for this condition's signal.*
- static FILE \* **lslogging\_file**

*our log file object*
- static lslogging\_queue\_t **lslogging\_queue** [LSLOGGING\_QUEUE\_LENGTH]
 

*Our entire queue. Right here. Every message we'll ever write.*
- static unsigned int **lslogging\_on** = 0
 

*next location to add to the queue*
- static unsigned int **lslogging\_off** = 0
 

*next location to remove from the queue*

### 7.3.1 Detailed Description

Logs messages to a file.

#### Date

2012

#### Author

Keith Brister

#### Copyright

All Rights Reserved

Definition in file [lslogging.c](#).

### 7.3.2 Macro Definition Documentation

#### 7.3.2.1 #define LSLOGGING\_FILE\_NAME "/tmp/pgpmac.log"

Full name of the log file.

Probably should be in /var/log/pgpmac.

Definition at line 16 of file [lslogging.c](#).

### 7.3.2.2 #define LSLOGGING\_MSG\_LENGTH 2048

Fixed maximum length messages to keep some form of sanity.

Definition at line 20 of file lslogging.c.

### 7.3.2.3 #define LSLOGGING\_QUEUE\_LENGTH 8192

Modest length queue.

Definition at line 30 of file lslogging.c.

## 7.3.3 Typedef Documentation

### 7.3.3.1 typedef struct lslogging\_queue\_struct lslogging\_queue\_t

Our log object: time and message.

## 7.3.4 Function Documentation

### 7.3.4.1 void lslogging\_event\_cb ( char \* event )

Log most events.

Definition at line 76 of file lslogging.c.

```

    {
if( strcmp( event, "Timer Update KVs") != 0 && strstr( event, "accepted") ==
    NULL && strstr( event, "queued") ==NULL) {
    lslogging_log_message( "EVENT: %s", event);
}
}
```

### 7.3.4.2 void lslogging\_init ( )

Initialize the lslogging objects.

Definition at line 37 of file lslogging.c.

```

    {
pthread_mutex_init( &lslogging_mutex, NULL);
pthread_cond_init( &lslogging_cond, NULL);

lslogging_file = fopen( LSLOGGING_FILE_NAME,
                      "w");
}
```

### 7.3.4.3 void lslogging\_log\_message ( char \* fmt, ... )

The routine everyone will be talking about.

#### Parameters

<i>fmt</i>	A printf style formating string.
...	The arguments specified by fmt

Definition at line 48 of file lslogging.c.

{

```

char msg[LSLOGGING_MSG_LENGTH];
struct timespec theTime;
va_list arg_ptr;
unsigned int on;

clock_gettime( CLOCK_REALTIME, &theTime);

va_start( arg_ptr, fmt);
vsnprintf( msg, sizeof(msg)-1, fmt, arg_ptr);
va_end( arg_ptr);
msg[sizeof(msg)-1]=0;

pthread_mutex_lock( &lslogging_mutex);

on = (lslogging_on++) % LSLOGGING_QUEUE_LENGTH
;
strncpy( lslogging_queue[on].lmsg, msg, LSLOGGING_MSG_LENGTH
- 1);
lslogging_queue[on].lmsg[LSLOGGING_MSG_LENGTH
-1] = 0;

memcpy( &(lslogging_queue[on].ltime), &theTime, sizeof(theTime
));

pthread_cond_signal( &lslogging_cond);
pthread_mutex_unlock( &lslogging_mutex);

}

```

#### 7.3.4.4 void lslogging\_run( )

Start up the worker thread.

Definition at line 121 of file lslogging.c.

```

{
pthread_create( &lslogging_thread, NULL, &lslogging_worker
, NULL);
lslogging_log_message( "Start up");
lsevents_add_listener( "+", lslogging_event_cb
);
}

```

#### 7.3.4.5 void\* lslogging\_worker( void \* dummy )

Service the queue, write to the file.

##### Parameters

in	<i>dummy</i>	Required by protocol but unused
----	--------------	---------------------------------

Definition at line 85 of file lslogging.c.

```

{

struct tm coarsetime;
char tstr[64];
unsigned int msecs;
unsigned int off;

pthread_mutex_lock( &lslogging_mutex);

while( 1) {
    while( lslogging_on == lslogging_off) {
        pthread_cond_wait( &lslogging_cond, &lslogging_mutex
    );
}

off = (lslogging_off++) % LSLOGGING_QUEUE_LENGTH
;

localtime_r( &(lslogging_queue[off].ltime.tv_sec), &
coarsetime);

```

```

strftime( tstr, sizeof(tstr)-1, "%Y-%m-%d %H:%M:%S", &coarsetime);
tstr[sizeof(tstr)-1] = 0;
msecs = lslogging_queue[off].ltime.tv_nsec / 1000;
fprintf( lslogging_file, "%s.%06u %s\n", tstr, msecs,
lslogging_queue[off].lmsg);
fflush( lslogging_file);

//
// If the newline comes after the string then only a blank line comes out
// in the ncurses terminal. Don't know why.
//
pgpmac_printf( "\n%s", lslogging_queue[off].
lmsg);
}
}

```

### 7.3.5 Variable Documentation

#### 7.3.5.1 pthread\_cond\_t lslogging\_cond [static]

We'll spend most of our time waiting for this condition's signal.

Definition at line 12 of file lslogging.c.

#### 7.3.5.2 FILE\* lslogging\_file [static]

our log file object

Definition at line 17 of file lslogging.c.

#### 7.3.5.3 pthread\_mutex\_t lslogging\_mutex [static]

mutex to keep the various threads from adding to the queue at the exact same time

Definition at line 11 of file lslogging.c.

#### 7.3.5.4 unsigned int lslogging\_off = 0 [static]

next location to remove from the queue

Definition at line 34 of file lslogging.c.

#### 7.3.5.5 unsigned int lslogging\_on = 0 [static]

next location to add to the queue

Definition at line 33 of file lslogging.c.

#### 7.3.5.6 lslogging\_queue\_t lslogging\_queue[LSLOGGING\_QUEUE\_LENGTH] [static]

Our entire queue. Right here. Every message we'll ever write.

Definition at line 31 of file lslogging.c.

#### 7.3.5.7 pthread\_t lslogging\_thread [static]

our thread

Definition at line 10 of file lslogging.c.

## 7.4 lspg.c File Reference

Postgresql support for the LS-CAT pgpmac project.

```
#include "pgpmac.h"
```

### Data Structures

- struct `lspg_wait_for_detector_struct`  
*Object that implements detector / spindle timing We use database locks for exposure control and this implements the md2 portion of this handshake.*
- struct `lspg_lock_diffractometer_struct`  
*Object used to implement locking the diffractometer Critical to exposure timing.*
- struct `lspg_lock_detector_struct`  
*lock detector object Implements detector lock for exposure control*
- struct `lspg_seq_run_prep_struct`  
*Data collection running object.*

### Macros

- #define `LS_PG_STATE_INIT` -4
- #define `LS_PG_STATE_INIT_POLL` -3
- #define `LS_PG_STATE_RESET` -2
- #define `LS_PG_STATE_RESET_POLL` -1
- #define `LS_PG_STATE_IDLE` 1
- #define `LS_PG_STATE_SEND` 2
- #define `LS_PG_STATE_SEND_FLUSH` 3
- #define `LS_PG_STATE_RECV` 4
- #define `LS_PG_QUERY_QUEUE_LENGTH` 16384  
*Queue length should be long enough that we do not ordinarily bump into the end We should be safe as long as the thread the adds stuff to the queue is not the one that removes it.*

### Typedefs

- typedef struct  
`lspg_wait_for_detector_struct` `lspg_wait_for_detector_t`  
*Object that implements detector / spindle timing We use database locks for exposure control and this implements the md2 portion of this handshake.*
- typedef struct  
`lspg_lock_diffractometer_struct` `lspg_lock_diffractometer_t`  
*Object used to implement locking the diffractometer Critical to exposure timing.*
- typedef struct  
`lspg_lock_detector_struct` `lspg_lock_detector_t`  
*lock detector object Implements detector lock for exposure control*
- typedef struct  
`lspg_seq_run_prep_struct` `lspg_seq_run_prep_t`  
*Data collection running object.*

## Functions

- `lspg_query_queue_t * lspg_query_next ()`  
*Return the next item in the postgresql queue.*
- `void lspg_query_reply_next ()`  
*Remove the oldest item in the queue.*
- `lspg_query_queue_t * lspg_query_reply_peek ()`  
*Return the next item in the reply queue but don't pop it since we may need it more than once.*
- `void lspg_query_push (void(*cb)(lspg_query_queue_t *, PGresult *), char *fmt,...)`  
*Place a query on the queue.*
- `char ** lspg_array2ptrs (char *a)`  
*returns a null terminated list of strings parsed from postgresql array*
- `void lspg_allkvs_cb (lspg_query_queue_t *qqp, PGresult *pgr)`  
*set a redis variable based on an updated kv pair*
- `void lspg_update_kvs_cb (char *event)`  
*Perhaps update the px.kvs table in postgresql Should be triggered by a timer event.*
- `void lspg_startransfer_init ()`
- `void lspg_startransfer_cb (lspg_query_queue_t *qqp, PGresult *pgr)`
- `void lspg_startransfer_call (unsigned int nextsample, int sample_detected, double ax, double ay, double az, double horz, double vert, double esttime)`
- `void lspg_startransfer_wait ()`
- `void lspg_startransfer_done ()`
- `int lspg_startransfer_all (int *err, unsigned int nextsample, int sample_detected, double ax, double ay, double az, double horz, double vert, double esttime)`
- `void lspg_getcurrentsampleid_init ()`
- `void lspg_getcurrentsampleid_cb (lspg_query_queue_t *qqp, PGresult *pgr)`  
*get currentsampleid*
- `void lspg_getcurrentsampleid_call ()`
- `unsigned int lspg_getcurrentsampleid_read ()`
- `void lspg_getcurrentsampleid_wait_for_id (unsigned int test)`
- `void lspg_nexsample_cb (lspg_query_queue_t *qqp, PGresult *pgr)`  
*Next Sample.*
- `void lspg_nexsample_init ()`  
*Initialize the nexsample variable, mutex, and condition.*
- `void lspg_nexsample_call ()`  
*Queue up a nexsample query.*
- `void lspg_nexsample_wait ()`  
*Wait for the nexsample query to get processed.*
- `void lspg_nexsample_done ()`  
*Called when the next shot query has been processed.*
- `unsigned int lspg_nexsample_all (int *err)`
- `void lspg_waitcryo_init ()`
- `void lspg_waitcryo_cb (lspg_query_queue_t *qqp, PGresult *pgr)`
- `void lspg_waitcryo_all ()`  
*no need to get fancy with the wait cryo command It should not return until the robot is almost ready for air rights*
- `void lspg_demandairrights_init ()`  
*initialize the demandairrights structure*
- `void lspg_demandairrights_cb (lspg_query_queue_t *qqp, PGresult *pgr)`  
*handle the airrights response*
- `void lspg_demandairrights_call ()`  
*call for airrights*
- `void lspg_demandairrights_wait ()`

- void **`lspg_demandairrights_all ()`**

*wait for the air rights request to return*  
*do nothing until we get airrights*
- void **`lspg_nextshot_cb (lspg_query_queue_t *qqp, PGresult *pgr)`**

*Next Shot Callback.*
- void **`lspg_nextshot_init ()`**

*Initialize the nextshot variable, mutex, and condition.*
- void **`lspg_nextshot_call ()`**

*Queue up a nextshot query.*
- void **`lspg_nextshot_wait ()`**

*Wait for the next shot query to get processed.*
- void **`lspg_nextshot_done ()`**

*Called when the next shot query has been processed.*
- void **`lspg_wait_for_detector_init ()`**

*initialize the detector timing object*
- void **`lspg_wait_for_detector_cb (lspg_query_queue_t *qqp, PGresult *pgr)`**

*Callback for the wait for detector query.*
- void **`lspg_wait_for_detector_call ()`**

*initiate the wait for detector query*
- void **`lspg_wait_for_detector_wait ()`**

*Pause the calling thread until the detector is ready Called by the MD2 thread.*
- void **`lspg_wait_for_detector_done ()`**

*Done waiting for the detector.*
- void **`lspg_wait_for_detector_all ()`**

*Combined call to wait for the detector.*
- void **`lspg_lock_diffractometer_init ()`**

*initialize the diffractometer locking object*
- void **`lspg_lock_diffractometer_cb (lspg_query_queue_t *qqp, PGresult *pgr)`**

*Callback routine for a lock diffractometer query.*
- void **`lspg_lock_diffractometer_call ()`**

*Request that the database grab the diffractometer lock.*
- void **`lspg_lock_diffractometer_wait ()`**

*Wait for the diffractometer lock.*
- void **`lspg_lock_diffractometer_done ()`**

*Finish up the lock diffractometer call.*
- void **`lspg_lock_diffractometer_all ()`**

*Convience function that combines lock diffractometer calls.*
- void **`lspg_lock_detector_init ()`**

*Initialize detector lock object.*
- void **`lspg_lock_detector_cb (lspg_query_queue_t *qqp, PGresult *pgr)`**

*Callback for when the detector lock has be grabbed.*
- void **`lspg_lock_detector_call ()`**

*Request (demand) a detector lock.*
- void **`lspg_lock_detector_wait ()`**

*Wait for the detector lock.*
- void **`lspg_lock_detector_done ()`**

*Finish waiting.*
- void **`lspg_lock_detector_all ()`**

*Detector lock convinience function.*
- void **`lspg_seq_run_prep_init ()`**

*Initialize the data collection object.*

- void `lspg_seq_run_prep_cb` (`lspg_query_queue_t` \*`qqp`, `PGresult` \*`pgr`)
 

*Callback for the seq\_run\_prep query.*
- void `lspg_seq_run_prep_call` (long long `skey`, double `kappa`, double `phi`, double `cx`, double `cy`, double `ax`, double `ay`, double `az`)
 

*queue up the seq\_run\_prep query*
- void `lspg_seq_run_prep_wait` ()
 

*Wait for seq run prep query to return.*
- void `lspg_seq_run_prep_done` ()
 

*Indicate we are done waiting.*
- void `lspg_seq_run_prep_all` (long long `skey`, double `kappa`, double `phi`, double `cx`, double `cy`, double `ax`, double `ay`, double `az`)
 

*Convinience function to call seq run prep.*
- void `lspg_getcenter_cb` (`lspg_query_queue_t` \*`qqp`, `PGresult` \*`pgr`)
 

*Retrieve the data to center the crystal.*
- void `lspg_getcenter_init` ()
 

*Initialize getcenter object.*
- void `lspg_getcenter_call` ()
 

*Request a getcenter query.*
- void `lspg_getcenter_wait` ()
 

*Wait for a getcenter query to return.*
- void `lspg_getcenter_done` ()
 

*Done with getcenter query.*
- void `lspg_getcenter_all` ()
 

*Convenience function to complete synchronous getcenter query.*
- void `lspg_nextaction_cb` (`lspg_query_queue_t` \*`qqp`, `PGresult` \*`pgr`)
 

*Queue the next MD2 instruction.*
- void `lspg_nexterrors_cb` (`lspg_query_queue_t` \*`qqp`, `PGresult` \*`pgr`)
 

*Send strings directly to PMAC queue.*
- void `lspg_cmd_cb` (`lspg_query_queue_t` \*`qqp`, `PGresult` \*`pgr`)
 

*Send strings directly to PMAC queue.*
- void `lspg_flush` ()
 

*Flush psql output buffer (ie, send the query)*
- void `lspg_send_next_query` ()
 

*send the next queued query to the DB server*
- void `lspg_receive` ()
 

*Receive a result of a query.*
- void `lspg_sig_service` (struct `pollfd` \*`evt`)
 

*Service a signal Signals here are treated as file descriptors and fits into our poll scheme.*
- void `lspg_pg_service` (struct `pollfd` \*`evt`)
 

*I/O control to/from the postgresql server.*
- PQnoticeProcessor `lspg_notice_processor` (void \*`arg`, const char \*`msg`)
 

`lspg_notice_processor`
- void `lspg_pg_connect` ()
 

*Connect to the pg server.*
- void `lspg_next_state` ()
 

*Implements our state machine Does not strictly only set the next state as it also calls some functions that, perhaps, alters the state mid-function.*
- void \* `lspg_worker` (void \*`dummy`)
 

*The main loop for the lspg thread.*
- void `lspg_preset_changed_cb` (char \*`event`)
 

`lspg_preset_changed_cb`
- void `lspg_check_preset_in_position_cb` (char \*`event`)
 

`lspg_check_preset_in_position_cb`
- void `lspg_unset_current_preset_moving_cb` (char \*`event`)
 

`lspg_unset_current_preset_moving_cb`
- void `lspg_set_scale_cb` (char \*`event`)
 

`lspg_set_scale_cb`

- void **lspg\_sample\_detector\_cb** (char \*event)
 

*Fix up xscale and yscale when zoom changes.*
- void **lspg\_quitting\_cb** (char \*event)
 

*log magnet state*
- void **lspg\_init ()**

*Prepare to exit the program in a couple of seconds.*
- void **lspg\_run ()**

*Initialize the lspg module.*
- void **lspg\_starttransfer\_t** **lspg\_starttransfer**

*Start 'er runnin'.*

## Variables

- static int **ls\_pg\_state** = LS\_PG\_STATE\_INIT
 

*State of the lspg state machine.*
- static struct timeval **lspg\_time\_sent now**

*used to ensure we do not inundate the db server with connection requests*
- static pthread\_t **lspg\_thread**

*our worker thread*
- static pthread\_mutex\_t **lspg\_queue\_mutex**

*keep the queue from getting tangled*
- static pthread\_cond\_t **lspg\_queue\_cond**

*keeps the queue from overflowing*
- static struct pollfd **lspgfd**

*our poll info*
- static **lspg\_query\_queue\_t** **lspg\_query\_queue** [LS\_PG\_QUERY\_QUEUE\_LENGTH]
 

*Our query queue.*
- static unsigned int **lspg\_query\_queue\_on** = 0
 

*Next position to add something to the queue.*
- static unsigned int **lspg\_query\_queue\_off** = 0
 

*The last item still being used (on == off means nothing in queue)*
- static unsigned int **lspg\_query\_queue\_reply** = 0
 

*The current item being digested.*
- static PGconn \* **q** = NULL
 

*Database connector.*
- static PostgresPollingStatusType **lspg\_connectPoll\_response**

*Used to determine state while connecting.*
- static PostgresPollingStatusType **lspg\_resetPoll\_response**

*Used to determine state while reconnecting.*
- **lspg\_nexsample\_t** **lspg\_nexsample**

*the very next sample*
- **lspg\_nextshot\_t** **lspg\_nextshot**

*the nextshot object*
- **lspg\_getcenter\_t** **lspg\_getcenter**

*the getcenter object*
- **lspg\_demandairrights\_t** **lspg\_demandairrights**

*our demandairrights object*
- **lspg\_getcurrentsampleid\_t** **lspg\_getcurrentsampleid**

*our currentsAMPLE id*
- **lspg\_starttransfer\_t** **lspg\_starttransfer**

- start a sample transfer*
- `lspg_waitcryo_t` `lspg_waitcryo`
- signal the robot*
- static `lspg_wait_for_detector_t` `lspg_wait_for_detector`
- Instance of the detector timing object.*
- static `lspg_lock_diffractometer_t` `lspg_lock_diffractometer`
- static `lspg_lock_detector_t` `lspg_lock_detector`
- static `lspg_seq_run_prep_t` `lspg_seq_run_prep`

#### 7.4.1 Detailed Description

Postgresql support for the LS-CAT pgpmac project.

```
\date 2012
\author Keith Brister
\copyright All Rights Reserved
```

Database state machine

State	Description
-4	Initiate connection
-3	Poll until connection initialization is complete
-2	Initiate reset
-1	Poll until connection reset is complete
1	Idle (wait for a notify from the server)
2	Send a query to the server
3	Continue flushing a command to the server
4	Waiting for a reply

Definition in file `lspg.c`.

#### 7.4.2 Macro Definition Documentation

##### 7.4.2.1 `#define LS_PG_QUERY_QUEUE_LENGTH 16384`

Queue length should be long enough that we do not ordinarily bump into the end We should be safe as long as the thread the adds stuff to the queue is not the one that removes it.

(And we can tolerate the adding thread being paused.)

Definition at line 50 of file `lspg.c`.

##### 7.4.2.2 `#define LS_PG_STATE_IDLE 1`

Definition at line 33 of file `lspg.c`.

##### 7.4.2.3 `#define LS_PG_STATE_INIT -4`

Definition at line 29 of file `lspg.c`.

##### 7.4.2.4 `#define LS_PG_STATE_INIT_POLL -3`

Definition at line 30 of file `lspg.c`.

## 7.4.2.5 #define LS\_PG\_STATE\_RECV 4

Definition at line 36 of file lspg.c.

## 7.4.2.6 #define LS\_PG\_STATE\_RESET -2

Definition at line 31 of file lspg.c.

## 7.4.2.7 #define LS\_PG\_STATE\_RESET\_POLL -1

Definition at line 32 of file lspg.c.

## 7.4.2.8 #define LS\_PG\_STATE\_SEND 2

Definition at line 34 of file lspg.c.

## 7.4.2.9 #define LS\_PG\_STATE\_SEND\_FLUSH 3

Definition at line 35 of file lspg.c.

## 7.4.3 Typedef Documentation

## 7.4.3.1 typedef struct lspg\_lock\_detector\_struct lspg\_lock\_detector\_t

lock detector object Implements detector lock for exposure control

## 7.4.3.2 typedef struct lspg\_lock\_diffractometer\_struct lspg\_lock\_diffractometer\_t

Object used to implement locking the diffractometer Critical to exposure timing.

## 7.4.3.3 typedef struct lspg\_seq\_run\_prep\_struct lspg\_seq\_run\_prep\_t

Data collection running object.

## 7.4.3.4 typedef struct lspg\_wait\_for\_detector\_struct lspg\_wait\_for\_detector\_t

Object that implements detector / spindle timing We use database locks for exposure control and this implements the md2 portion of this handshake.

## 7.4.4 Function Documentation

## 7.4.4.1 void lspg\_allkvs\_cb ( lspg\_query\_queue\_t \* qqp, PGresult \* pgr )

set a redis variable based on an updated kv pair

## Parameters

<i>qqp</i>	The query that elicited this response
<i>pgr</i>	The response from postgresql

Definition at line 279 of file lspg.c.

```

        {
int i;
_lsredis_obj_t *robj;

for( i=0; i<PQntuples( pgr); i += 2) {
    pthread_mutex_lock( &lsredis_mutex);
    while( lsredis_running == 0)
        pthread_cond_wait( &lsredis_cond, &lsredis_mutex
    );
    pthread_mutex_unlock( &lsredis_mutex);

    robj = _lsredis_get_obj( PQgetvalue( pgr, i, 0));

    if( robj == NULL) {
        lslogging_log_message( "lspg_allkvs_cb: could not
            find redis object named '%s'", PQgetvalue( pgr, i, 0));
        continue;
    }

    lsredis_setstr( robj, "%s", PQgetvalue( pgr, i+1, 0));
}
}

```

#### 7.4.4.2 char\*\* lspg\_array2ptrs ( char \* a )

returns a null terminated list of strings parsed from postgresql array

Definition at line 160 of file lspg.c.

```

{
char **rtn, *sp, *acums;
int i, n, inquote, havebackslash, rtni;;
int mxsz;

inquote      = 0;
havebackslash = 0;

// Despense with the null input condition before we complicate the code below
if( a == NULL || a[0] != '{' || a[strlen(a)-1] != '}')
    return NULL;

// Count the maximum number of strings
// Actual number will be less if there are quoted commas
//
n = 1;
for( i=0; a[i]; i++) {
    if( a[i] == ',')
        n++;
}
//
// The maximum size of any string is the length of a (+1)
//
mxsz = strlen(a) + 1;

// This is the accumulation string to make up the array elements
acums = (char *)calloc( mxsz, sizeof( char));
if( acums == NULL) {
    lslogging_log_message( "lspg_array2ptrs: out of memory
        (acums)");
    exit( 1);
}

//
// allocate storage for the pointer array and the null terminator
//
rtn = (char **)calloc( n+1, sizeof( char *));
if( rtn == NULL) {
    lslogging_log_message( "lspg_array2ptrs: out of memory
        (rtn)");
    exit( 1);
}
rtni = 0;

// Go through and create the individual strings
sp = acums;
*sp = 0;

inquote = 0;
havebackslash = 0;
for( i=1; a[i] != 0; i++) {
    switch( a[i]) {

```

```

    case '\"':
        if( havebackslash) {
            // a quoted quote.  Cool
            //
            *(sp++) = a[i];
            *sp = 0;
            havebackslash = 0;
        } else {
            // Toggle the flag
            inquote = 1 - inquote;
        }
        break;

    case '\\':
        if( havebackslash) {
            *(sp++) = a[i];
            *sp = 0;
            havebackslash = 0;
        } else {
            havebackslash = 1;
        }
        break;

    case ',':
        if( inquote || havebackslash) {
            *(sp++) = a[i];
            *sp = 0;
            havebackslash = 0;
        } else {
            rtn[rtni++] = strdup( acums);
            sp = acums;
        }
        break;

    case ')':
        if( inquote || havebackslash) {
            *(sp++) = a[i];
            *sp = 0;
            havebackslash = 0;
        } else {
            rtn[rtni++] = strdup( acums);
            rtn[rtni] = NULL;
            free( acums);
            return( rtn);
        }
        break;

    default:
        *(sp++) = a[i];
        *sp = 0;
        havebackslash = 0;
    }
}

// Getting here means the final ')' was missing
// Probably we should throw an error or log it or something.
// Through out the last entry since this there is not resonable expectation
// that
// we should be parsing it anyway.
//
rtn[rtni] = NULL;
free( acums);
return( rtn);
}

```

#### 7.4.4.3 void lspg\_check\_preset\_in\_position\_cb ( char \* event )

Definition at line 1890 of file lspg.c.

```

{
lspmac_motor_t *mp;
char cp[64];
int i;

for( i=0; i<strlen( event); i++) {
    cp[i] = 0;
    if( event[i] == ' ')
        break;
    cp[i] = event[i];
}

mp = lspmac_find_motor_by_name( cp);

```

```

if( mp == NULL) {
    return;
}
i = lsredis_find_preset_index_by_position
    ( mp);
lspg_query_push( NULL, "EXECUTE kvupdate(
    '%s.currentPreset,%d')", cp, i);

}

```

#### 7.4.4.4 void lspg\_cmd\_cb ( lspg\_query\_queue\_t \* qqp, PGresult \* pgr )

Send strings directly to PMAC queue.

##### Parameters

in	qqp	Our query
in	pgr	Our result

Definition at line 1354 of file lspg.c.

```

{
// Call back funciton assumes query results in zero or more commands to send
// to the PMAC
//
int i;
char *sp;

for( i=0; i<PQntuples( pgr); i++) {
    sp = PQgetvalue( pgr, i, 0);
    if( sp != NULL && *sp != 0) {
        lspmacc_SockSendDPLine( NULL, sp);
        // lspmacc_SockSendline( sp);
        //
        // Keep asking for more until
        // there are no commands left
        //
        // This should solve a potential problem where
        // more than one command is put on the queue for a given notify.
        //
        lspg_query_push( lspg_cmd_cb, "select
            pmac.md2_queue_next()");

    }
}
}
```

#### 7.4.4.5 void lspg\_demandairrights\_all( )

do nothing until we get airrights

Definition at line 647 of file lspg.c.

```

{
lspg_demandairrights_call();
lspg_demandairrights_wait();
// there is no "done" version
}
```

#### 7.4.4.6 void lspg\_demandairrights\_call( )

call for airrights

Definition at line 629 of file lspg.c.

```

{
pthread_mutex_lock( &lspg_demandairrights.mutex);
lspg_demandairrights.new_value_ready = 0;
```

```

pthread_mutex_unlock( &lspg_demandairrights.mutex);
lspg_query_push( lspg_demandairrights_cb
    , "SELECT px.demandairrights()");
}

```

#### 7.4.4.7 void lspg\_demandairrights\_cb ( lspg\_query\_queue\_t \* qqp, PGresult \* pgr )

handle the airrights response

Definition at line 619 of file lspg.c.

```

{
pthread_mutex_lock( &lspg_demandairrights.mutex);
lspg_demandairrights.new_value_ready = 1;
pthread_cond_signal( &lspg_demandairrights.cond);
pthread_mutex_unlock( &lspg_demandairrights.mutex);
lslogging_log_message( "lspg_demandairrights_cb: Here I
am");
}

```

#### 7.4.4.8 void lspg\_demandairrights\_init ( )

initialize the demandairrights structure

Definition at line 611 of file lspg.c.

```

{
lspg_demandairrights.new_value_ready = 0;
pthread_mutex_init( &lspg_demandairrights.mutex,
    NULL);
pthread_cond_init( &lspg_demandairrights.cond, NULL);
}

```

#### 7.4.4.9 void lspg\_demandairrights\_wait ( )

wait for the air rights request to return

Definition at line 638 of file lspg.c.

```

{
pthread_mutex_lock( &lspg_demandairrights.mutex);
while( lspg_demandairrights.new_value_ready
    == 0)
    pthread_cond_wait( &lspg_demandairrights.cond, &
        lspg_demandairrights.mutex);
pthread_mutex_unlock( &lspg_demandairrights.mutex);
}

```

#### 7.4.4.10 void lspg\_flush ( )

Flush psql output buffer (ie, send the query)

Definition at line 1384 of file lspg.c.

```

{
int err;

err = PQflush( q);
switch( err) {
case -1:
    // an error occurred

    lslogging_log_message( "flush failed: %s",
        PQerrorMessage( q));
}

```

```

ls_pg_state = LS_PG_STATE_IDLE;
//
// We should probably reset the connection and start from scratch.
// Probably the connection died.
//
break;

case 0:
// goodness and joy.
ls_pg_state = LS_PG_STATE_RECV;
break;

case 1:
// more sending to do
ls_pg_state = LS_PG_STATE_SEND_FLUSH;
break;
}
}

```

#### 7.4.4.11 void lsgc\_getcenter\_all( )

Convenience function to complete synchronous getcenter query.

Definition at line 1292 of file lsgc.c.

```

{
lsgc_getcenter_call();
lsgc_getcenter_wait();
lsgc_getcenter_done();
}

```

#### 7.4.4.12 void lsgc\_getcenter\_call( )

Request a getcenter query.

Definition at line 1268 of file lsgc.c.

```

{
pthread_mutex_lock( &lsgc_getcenter.mutex );
lsgc_getcenter.new_value_ready = 0;
pthread_mutex_unlock( &lsgc_getcenter.mutex );

lsgc_query_push( lsgc_getcenter_cb, "SELECT * "
                  "FROM px.getcenter2() ");
}

```

#### 7.4.4.13 void lsgc\_getcenter\_cb( lsgc\_query\_queue\_t \* qqp, PGresult \* pgr )

Retrieve the data to center the crystal.

Definition at line 1203 of file lsgc.c.

```

{
static int
zoom_c, dcx_c, dcy_c, dax_c, day_c, daz_c;

pthread_mutex_lock( &(lsgc_getcenter.mutex) );

lsgc_getcenter.no_rows_returned = PQntuples(
    pgr) <= 0;
if( lsgc_getcenter.no_rows_returned ) {
//
// No particular reason this path should ever be taken
// but if we don't get rows then we had better not move anything.
//
lsgc_getcenter.new_value_ready = 1;
pthread_cond_signal( &(lsgc_getcenter.cond));
pthread_mutex_unlock( &(lsgc_getcenter.mutex));
return;
}

```

```

zoom_c = PQfnumber( pgr, "zoom");
dcx_c = PQfnumber( pgr, "dcx");
dcy_c = PQfnumber( pgr, "dcy");
dax_c = PQfnumber( pgr, "dax");
day_c = PQfnumber( pgr, "day");
daz_c = PQfnumber( pgr, "daz");

lspg_getcenter.zoom_isnull = PQgetisnull( pgr, 0,
    zoom_c);
if( lspg_getcenter.zoom_isnull == 0)
    lspg_getcenter.zoom = atoi( PQgetvalue( pgr, 0, zoom_c));

lspg_getcenter.dcx_isnull = PQgetisnull( pgr, 0,
    dcx_c);
if( lspg_getcenter.dcx_isnull == 0)
    lspg_getcenter.dcx = atof( PQgetvalue( pgr, 0, dcx_c));

lspg_getcenter.dcy_isnull = PQgetisnull( pgr, 0,
    dcy_c);
if( lspg_getcenter.dcy_isnull == 0)
    lspg_getcenter.dcy = atof( PQgetvalue( pgr, 0, dcy_c));

lspg_getcenter.dax_isnull = PQgetisnull( pgr, 0,
    dax_c);
if( lspg_getcenter.dax_isnull == 0)
    lspg_getcenter.dax = atof( PQgetvalue( pgr, 0, dax_c));

lspg_getcenter.day_isnull = PQgetisnull( pgr, 0,
    day_c);
if( lspg_getcenter.day_isnull == 0)
    lspg_getcenter.day = atof( PQgetvalue( pgr, 0, day_c));

lspg_getcenter.daz_isnull = PQgetisnull( pgr, 0,
    daz_c);
if( lspg_getcenter.daz_isnull == 0)
    lspg_getcenter.daz = atof( PQgetvalue( pgr, 0, daz_c));

lspg_getcenter.new_value_ready = 1;

pthread_cond_signal( &(lspg_getcenter.cond));
pthread_mutex_unlock( &(lspg_getcenter.mutex));
}

```

#### 7.4.4.14 void lspg\_getcenter\_done( )

Done with getcenter query.

Definition at line 1286 of file lspg.c.

```

{
    pthread_mutex_unlock( &(lspg_getcenter.mutex));
}

```

#### 7.4.4.15 void lspg\_getcenter\_init( )

Initialize getcenter object.

Definition at line 1260 of file lspg.c.

```

{
    memset( &lspg_getcenter, 0, sizeof( lspg_getcenter
        ));
    pthread_mutex_init( &(lspg_getcenter.mutex), NULL);
    pthread_cond_init( &(lspg_getcenter.cond), NULL);
}

```

#### 7.4.4.16 void lspg\_getcenter\_wait( )

Wait for a getcenter query to return.

Definition at line 1278 of file lspg.c.

```

    {
        pthread_mutex_lock( &(lspg_getcenter.mutex));
        while( lspg_getcenter.new_value_ready == 0)
            pthread_cond_wait( &(lspg_getcenter.cond), &
                lspg_getcenter.mutex));
    }
}

```

#### 7.4.4.17 void lspg\_getcurrentsampleid\_call( )

Definition at line 457 of file Lspg.c.

```

    {
        pthread_mutex_lock( &lspg_getcurrentsampleid.mutex
        );
        lspg_getcurrentsampleid.new_value_ready
        = 0;
        pthread_mutex_unlock( &lspg_getcurrentsampleid.mutex
        );

        lspg_query_push( lspg_getcurrentsampleid_cb
            , "SELECT px.getcurrentsampleid()");
    }
}

```

#### 7.4.4.18 void lspg\_getcurrentsampleid\_cb( Lspg\_query\_queue\_t \* qqp, PGresult \* pgr )

get currentsampleid

Definition at line 433 of file Lspg.c.

```

    {
        pthread_mutex_lock( &lspg_getcurrentsampleid.mutex
        );

        lspg_nexsample.new_value_ready = 1;
        lspg_getcurrentsampleid.no_rows_returned
        = PQntuples( pgr) <= 0;
        if( lspg_getcurrentsampleid.no_rows_returned
            ) {
            pthread_cond_signal( &lspg_getcurrentsampleid.cond
            );
            pthread_mutex_unlock( &lspg_getcurrentsampleid.mutex
            );
            return;
        }

        lspg_getcurrentsampleid.getcurrentsampleid_isnull
        = PQgetisnull( pgr, 0, 0);
        if( lspg_getcurrentsampleid.getcurrentsampleid_isnull
            == 0)
            lspg_getcurrentsampleid.getcurrentsampleid
            = strtol( PQgetvalue( pgr, 0, 0), NULL, 0);

        lslogging_log_message( "lspg_getcurrentsampleid_cb:
            current sample id: %d",
            lspg_getcurrentsampleid.
            getcurrentsampleid);

        pthread_cond_signal( &lspg_getcurrentsampleid.cond
        );
        pthread_mutex_unlock( &lspg_getcurrentsampleid.mutex
        );
    }
}

```

#### 7.4.4.19 void lspg\_getcurrentsampleid\_init( )

Definition at line 425 of file Lspg.c.

```

    {
        lspg_getcurrentsampleid.new_value_ready
        = 0;
        pthread_mutex_init( &lspg_getcurrentsampleid.mutex

```

```

    , NULL);
pthread_cond_init( &lspg_getcurrentsampleid.cond,
                  NULL);
}

```

#### 7.4.4.20 unsigned int lspg\_getcurrentsampleid.read( )

Definition at line 467 of file lspg.c.

```

{
unsigned int rtn;
pthread_mutex_lock( &lspg_getcurrentsampleid.mutex
                   );
while( lspg_getcurrentsampleid.new_value_ready
      == 0)
pthread_cond_wait( &lspg_getcurrentsampleid.cond
                  , &lspg_getcurrentsampleid.mutex);
if( lspg_getcurrentsampleid.getcurrentsampleid_isnull
    )
rtn = -1;
else
rtn = lspg_getcurrentsampleid.getcurrentsampleid
      ;
pthread_mutex_unlock( &lspg_getcurrentsampleid.mutex
                     );
return rtn;
}

```

#### 7.4.4.21 void lspg\_getcurrentsampleid.wait\_for\_id( unsigned int test )

Definition at line 483 of file lspg.c.

```

{
pthread_mutex_lock( &lspg_getcurrentsampleid.mutex
                   );
while( lspg_getcurrentsampleid.getcurrentsampleid
      != test)
pthread_cond_wait( &lspg_getcurrentsampleid.cond
                  , &lspg_getcurrentsampleid.mutex);
pthread_mutex_unlock( &lspg_getcurrentsampleid.mutex
                     );
}

```

#### 7.4.4.22 void lspg\_init( )

Initialize the lspg module.

Definition at line 1979 of file lspg.c.

```

{
pthread_mutex_init( &lspg_queue_mutex, NULL);
pthread_cond_init( &lspg_queue_cond, NULL);

lspg_demandairrights_init();
lspg_getcenter_init();
lspg_getcurrentsampleid_init();
lspg_lock_detector_init();
lspg_lock_diffractometer_init();
lspg_nexsample_init();
lspg_nextshot_init();
lspg_seq_run_prep_init();
lspg_starttransfer_init();
lspg_wait_for_detector_init();
lspg_waitcryo_init();
}

```

#### 7.4.4.23 void lsgp\_lock\_detector\_all( )

Detector lock convinience function.

Definition at line 1115 of file lsgp.c.

```
{
lsgp_lock_detector_call();
lsgp_lock_detector_wait();
lsgp_lock_detector_done();
}
```

#### 7.4.4.24 void lsgp\_lock\_detector\_call( )

Request (demand) a detector lock.

Definition at line 1091 of file lsgp.c.

```
{
pthread_mutex_lock( &(lsgp_lock_detector.mutex));
lsgp_lock_detector.new_value_ready = 0;
pthread_mutex_unlock( &(lsgp_lock_detector.mutex));

lsgp_query_push( lsgp_lock_detector_cb,
    "SELECT px.lock_detector()");
}
```

#### 7.4.4.25 void lsgp\_lock\_detector\_cb( lsgp\_query\_queue\_t \* qqp, PGresult \* pgr )

Callback for when the detector lock has be grabbed.

Definition at line 1082 of file lsgp.c.

```
{
pthread_mutex_lock( &(lsgp_lock_detector.mutex));
lsgp_lock_detector.new_value_ready = 1;
pthread_cond_signal( &(lsgp_lock_detector.cond));
pthread_mutex_unlock( &(lsgp_lock_detector.mutex));
}
```

#### 7.4.4.26 void lsgp\_lock\_detector\_done( )

Finish waiting.

Definition at line 1109 of file lsgp.c.

```
{
pthread_mutex_unlock( &(lsgp_lock_detector.mutex));
}
```

#### 7.4.4.27 void lsgp\_lock\_detector\_init( )

Initialize detector lock object.

Definition at line 1074 of file lsgp.c.

```
{
lsgp_lock_detector.new_value_ready = 0;
pthread_mutex_init( &(lsgp_lock_detector.mutex), NULL)
;
pthread_cond_init( &(lsgp_lock_detector.cond), NULL);
}
```

## 7.4.4.28 void lspg\_lock\_detector\_wait( )

Wait for the detector lock.

Definition at line 1101 of file lspg.c.

```
{
pthread_mutex_lock( &(lspg_lock_detector.mutex));
while( lspg_lock_detector.new_value_ready ==
      0)
pthread_cond_wait( &(lspg_lock_detector.cond), &
                  lspg_lock_detector.mutex));
}
```

## 7.4.4.29 void lspg\_lock\_diffractometer\_all( )

Convience function that combines lock diffractometer calls.

Definition at line 1056 of file lspg.c.

```
{
lspg_lock_diffractometer_call();
lspg_lock_diffractometer_wait();
lspg_lock_diffractometer_all();
}
```

## 7.4.4.30 void lspg\_lock\_diffractometer\_call( )

Request that the database grab the diffractometer lock.

Definition at line 1032 of file lspg.c.

```
{
pthread_mutex_lock( &(lspg_lock_diffractometer.mutex
));
lspg_lock_diffractometer.new_value_ready
= 0;
pthread_mutex_unlock( &(lspg_lock_diffractometer.
                      mutex));
lspg_query_push( lspg_lock_diffractometer_cb
                , "SELECT px.lock_diffractometer()");
}
```

## 7.4.4.31 void lspg\_lock\_diffractometer\_cb( lspg\_query\_queue\_t \* qqp, PGresult \* pgr )

Callback routine for a lock diffractometer query.

Definition at line 1023 of file lspg.c.

```
{
pthread_mutex_lock( &(lspg_lock_diffractometer.mutex
));
lspg_lock_diffractometer.new_value_ready
= 1;
pthread_cond_signal( &(lspg_lock_diffractometer.cond
));
pthread_mutex_unlock( &(lspg_lock_diffractometer.
                      mutex));
}
```

#### 7.4.4.32 void lsgp\_lock\_diffractometer\_done( )

Finish up the lock diffractometer call.

Definition at line 1050 of file lsgp.c.

```
{
    pthread_mutex_unlock( &(lsgp_lock_diffractometer.
        mutex));
}
```

#### 7.4.4.33 void lsgp\_lock\_diffractometer\_init( )

initialize the diffractometer locking object

Definition at line 1015 of file lsgp.c.

```
{
    lsgp_lock_diffractometer.new_value_ready
        = 0;
    pthread_mutex_init( &(lsgp_lock_diffractometer.mutex
        ), NULL);
    pthread_cond_init( &(lsgp_lock_diffractometer.cond
        ), NULL);
}
```

#### 7.4.4.34 void lsgp\_lock\_diffractometer\_wait( )

Wait for the diffractometer lock.

Definition at line 1042 of file lsgp.c.

```
{
    pthread_mutex_lock( &(lsgp_lock_diffractometer.mutex
        ));
    while( lsgp_lock_diffractometer.new_value_ready
        == 0)
        pthread_cond_wait( &(lsgp_lock_diffractometer.cond
            ), &(lsgp_lock_diffractometer.mutex));
}
```

#### 7.4.4.35 void lsgp\_next\_state( )

Implements our state machine Does not strictly only set the next state as it also calls some functions that, perhaps, alters the state mid-function.

Definition at line 1743 of file lsgp.c.

```
{
    // connect to the database
    //
    if( q == NULL ||
        ls_pg_state == LS_PG_STATE_INIT ||
        ls_pg_state == LS_PG_STATE_RESET ||
        ls_pg_state == LS_PG_STATE_INIT_POLL ||
        ls_pg_state == LS_PG_STATE_RESET_POLL)
        lsgp_pg_connect( lsgpf);

    if( ls_pg_state == LS_PG_STATE_IDLE &&
        lsgp_query_queue_on != lsgp_query_queue_off
        )
        ls_pg_state = LS_PG_STATE_SEND;

    switch( ls_pg_state) {
        case LS_PG_STATE_INIT_POLL:
            if( lsgp_connectPoll_response ==

```

```

    PGRES_POLLING_WRITING)
    lspgfd.events = POLLOUT;
else if( lspg_connectPoll_response ==
    PGRES_POLLING_READING)
    lspgfd.events = POLLIN;
else
    lspgfd.events = 0;
break;

case LS_PG_STATE_RESET_POLL:
    if( lspg_resetPoll_response == PGRES_POLLING_WRITING
    )
        lspgfd.events = POLLOUT;
else if( lspg_resetPoll_response ==
    PGRES_POLLING_READING)
    lspgfd.events = POLLIN;
else
    lspgfd.events = 0;
break;

case LS_PG_STATE_IDLE:
case LS_PG_STATE_RECV:
    lspgfd.events = POLLIN;
break;

case LS_PG_STATE_SEND:
case LS_PG_STATE_SEND_FLUSH:
    lspgfd.events = POLLOUT;
break;

default:
    lspgfd.events = 0;
}
}

```

#### 7.4.4.36 void lspg\_nextaction\_cb ( Ispg\_query\_queue\_t \* qqp, PGresult \* pgr )

Queue the next MD2 instruction.

##### Parameters

in	qqp	The query that generated this result
in	pgr	The result

Definition at line 1301 of file lspg.c.

```

{
char *action;

if( PQntuples( pgr ) <= 0)
    return;           // Note: nextaction should always return at least
                      "noAction", so this branch should never be taken

action = PQgetvalue( pgr, 0, 0);      // next action only returns one row

if( strcmp( action, "noAction" ) == 0)
    return;

md2cmds_push_queue( action);

}

```

#### 7.4.4.37 void lspg\_nexterrors\_cb ( Ispg\_query\_queue\_t \* qqp, PGresult \* pgr )

Definition at line 1320 of file lspg.c.

```

{
static int etid_col, etseverity_col, etterse_col, etverbose_col,
    etdetails_col;
static int first_time=1;
int i;
char *terse, *verbose, *details, *severity, *id;

```

```

if( first_time) {
    etid_col      = PQfnumber( pgr, "etid");
    etseverity_col = PQfnumber( pgr, "etseverity");
    etterse_col   = PQfnumber( pgr, "etterse");
    etverbose_col = PQfnumber( pgr, "etverbose");
    etdetails_col = PQfnumber( pgr, "etdetails");
    first_time    = 0;
}

for( i=0; i<PQntuples( pgr); i++) {
    id       = PQgetvalue( pgr, i, etid_col);
    terse   = PQgetvalue( pgr, i, etterse_col);
    verbose = PQgetvalue( pgr, i, etverbose_col);
    details = PQgetvalue( pgr, i, etdetails_col);
    severity = PQgetvalue( pgr, i, etseverity_col);

    lspg_query_push( NULL, "EXECUTE acknowledgeerror(%s)", id);

    lslogging_log_message( "lspg_nexterrors_cb: %s %s\n",
        severity, strlen(verbose)>0 ? verbose : terse);
    if( strlen( details) > 0)
        lslogging_log_message( "lspg_nexterrors_cb: %s\n",
            details);
}
}
}

```

#### 7.4.4.38 unsigned int lspg\_nexsample\_all( int \*err )

Definition at line 558 of file `lspg.c`.

```

{
unsigned int rtn;

lspg_nexsample_call();
lspg_nexsample_wait();

if( lspg_nexsample.no_rows_returned) {
    rtn = 0;
    *err = 1;
} else {
    if( lspg_nexsample.nexsample_isnull) {
        rtn = 0;
        *err = 1;
    } else {
        rtn = lspg_nexsample.nexsample;
        *err = 0;
    }
}
lspg_nexsample_done();

return rtn;
}
}
```

#### 7.4.4.39 void lspg\_nexsample\_call( )

Queue up a nexsample query.

Definition at line 535 of file `lspg.c`.

```

{
pthread_mutex_lock( &(lspg_nexsample.mutex));
lspg_nexsample.new_value_ready = 0;
pthread_mutex_unlock( &(lspg_nexsample.mutex));

lspg_query_push( lspg_nexsample_cb, "SELECT
    nexsample FROM px.nexsample()");
}
}
```

#### 7.4.4.40 void lspg\_nexsample\_cb( lspg\_query\_queue\_t \*qqp, PGresult \*pgr )

Next Sample.

**Parameters**

in	<i>qqp</i>	Our nextsample query
in	<i>pgr</i>	result of the query

Definition at line 494 of file lspg.c.

```

{
static int got_columns = 0;
static int nextsample_col;
pthread_mutex_lock( &(lspg_nextsample.mutex));

lspg_nextsample.no_rows_returned = PQntuples(
    pgr) <= 0;
if( lspg_nextsample.no_rows_returned) {
    lslogging_log_message( "lspg_nextsample_cb: no rows
        returned. This should never happen.");
    lspg_nextsample.new_value_ready = 1;
    pthread_cond_signal( &(lspg_nextsample.cond));
    pthread_mutex_unlock( &(lspg_nextsample.mutex));
    return;
}

if( got_columns == 0) {
    nextsample_col = PQfnumber( pgr, "nextsample");
    got_columns = 1;
}

lspg_nextsample.nextsample_isnull =
    PQgetisnull( pgr, 0, nextsample_col);
if( lspg_nextsample.nextsample_isnull == 0)
    lspg_nextsample.nextsample = strtol( PQgetvalue(
        pgr, 0, nextsample_col), NULL, 0);

lspg_nextsample.new_value_ready = 1;
pthread_cond_signal( &(lspg_nextsample.cond));
pthread_mutex_unlock( &(lspg_nextsample.mutex));
}
}
```

**7.4.4.41 void lspg\_nextsample\_done( )**

Called when the next shot query has been processed.

Definition at line 553 of file lspg.c.

```

{
    pthread_mutex_unlock( &(lspg_nextsample.mutex));
}
```

**7.4.4.42 void lspg\_nextsample\_init( )**

Initialize the nextsample variable, mutex, and condition.

Definition at line 527 of file lspg.c.

```

{
memset( &lspg_nextsample, 0, sizeof( lspg_nextsample
    ));
pthread_mutex_init( &(lspg_nextsample.mutex), NULL);
pthread_cond_init( &(lspg_nextsample.cond), NULL);
}
```

**7.4.4.43 void lspg\_nextsample\_wait( )**

Wait for the nextsample query to get processed.

Definition at line 545 of file lspg.c.

```

    {
    pthread_mutex_lock( &(lspg_nextsample.mutex));
    while( lspg_nextsample.new_value_ready == 0)
        pthread_cond_wait( &(lspg_nextsample.cond), &
                           lspg_nextsample.mutex));
}

```

#### 7.4.4.44 void lspg\_nextshot\_call( )

Queue up a nextshot query.

Definition at line 915 of file lspg.c.

```

    {
    pthread_mutex_lock( &(lspg_nextshot.mutex));
    lspg_nextshot.new_value_ready = 0;
    pthread_mutex_unlock( &(lspg_nextshot.mutex));

    lspg_query_push( lspg_nextshot_cb, "SELECT *
                                FROM px.nextshot2()");
}

```

#### 7.4.4.45 void lspg\_nextshot\_cb ( lspg\_query\_queue\_t \* qqp, PGresult \* pgr )

Next Shot Callback.

This is a long and tedious routine as there are a large number of variables returned. Suck it up. Return with the global object lspg\_nextshot set.

##### Parameters

in	qqp	Our nextshot query
in	pgr	result of the query

Definition at line 660 of file lspg.c.

```

    {
static int got_col_nums=0;
static int
dsdir_c, dspid_c, dsowidth_c, dsoscaaxis_c, dsexp_c, skey_c, sstart_c, sfn_c
, dphi_c,
dsomega_c, dskappa_c, dsdist_c, dsnrg_c, dshpid_c, cx_c, cy_c, ax_c, ay_c,
az_c,
active_c, sindex_c, stype_c,
dsowidth2_c, dsoscaaxis2_c, dsexp2_c, sstart2_c, dphi2_c, dsomega2_c,
dskappa2_c, dsdist2_c, dsnrg2_c,
cx2_c, cy2_c, ax2_c, ay2_c, az2_c, active2_c, sindex2_c, stype2_c;

pthread_mutex_lock( &(lspg_nextshot.mutex));

lspg_nextshot.no_rows_returned = PQntuples( pgr)
    <= 0;
if( lspg_nextshot.no_rows_returned) {
    lspg_nextshot.new_value_ready = 1;
    pthread_cond_signal( &(lspg_nextshot.cond));
    pthread_mutex_unlock( &(lspg_nextshot.mutex));
    return; // I guess there was no shot after all
}

if( got_col_nums == 0) {
    dsdir_c = PQfnumber( pgr, "dsdir");
    dpid_c = PQfnumber( pgr, "dspid");
    dsowidth_c = PQfnumber( pgr, "dsowidth");
    dsoscaaxis_c = PQfnumber( pgr, "dsoscaaxis");
    dsexp_c = PQfnumber( pgr, "dsexp");
    skey_c = PQfnumber( pgr, "skey");
    sstart_c = PQfnumber( pgr, "sstart");
    sfn_c = PQfnumber( pgr, "sfm");
    dphi_c = PQfnumber( pgr, "dphi");
    dsomega_c = PQfnumber( pgr, "dsomega");
    dskappa_c = PQfnumber( pgr, "dskappa");
    dsdist_c = PQfnumber( pgr, "dsdist");
    dsnrg_c = PQfnumber( pgr, "dsnrg");
}

```

```

dshpid_c      = PQfnumber( pgr, "dshpid");
cx_c          = PQfnumber( pgr, "cx");
cy_c          = PQfnumber( pgr, "cy");
ax_c          = PQfnumber( pgr, "ax");
ay_c          = PQfnumber( pgr, "ay");
az_c          = PQfnumber( pgr, "az");
active_c       = PQfnumber( pgr, "active");
sindex_c       = PQfnumber( pgr, "sindex");
stype_c        = PQfnumber( pgr, "stype");
dsowidth2_c   = PQfnumber( pgr, "dsowidth2");
dsoscaxis2_c  = PQfnumber( pgr, "dsoscaxis2");
dsexp2_c       = PQfnumber( pgr, "dsexp2");
sstart2_c      = PQfnumber( pgr, "sstart2");
dsphi2_c       = PQfnumber( pgr, "dsphi2");
dsomega2_c     = PQfnumber( pgr, "dsomega2");
dskappa2_c    = PQfnumber( pgr, "dskappa2");
dsdist2_c     = PQfnumber( pgr, "dsdist2");
dsnrg2_c       = PQfnumber( pgr, "dsnrg2");
cx2_c          = PQfnumber( pgr, "cx2");
cy2_c          = PQfnumber( pgr, "cy2");
ax2_c          = PQfnumber( pgr, "ax2");
ay2_c          = PQfnumber( pgr, "ay2");
az2_c          = PQfnumber( pgr, "az2");
active2_c       = PQfnumber( pgr, "active2");
sindex2_c      = PQfnumber( pgr, "sindex2");
stype2_c        = PQfnumber( pgr, "stype2");

got_col_nums = 1;
}

//  

// NULL string values come back as empty strings  

// Mark the null flag but allocate the empty string anyway  

//  

lspg_nextshot.dsdir_isnull = PQgetisnull( pgr, 0,  

    dsdir_c);  

if( lspg_nextshot.dsdir != NULL)  

    free( lspg_nextshot.dsdir);  

lspg_nextshot.dsdir = strdup( PQgetvalue( pgr, 0, dsdir_c))  

;  

lspg_nextshot.dspid_isnull = PQgetisnull( pgr, 0,  

    dpid_c);  

if( lspg_nextshot.dpid != NULL)  

    free( lspg_nextshot.dpid);  

lspg_nextshot.dpid = strdup( PQgetvalue( pgr, 0, dpid_c))  

;  

lspg_nextshot.dsoscaxis_isnull = PQgetisnull(  

    pgr, 0, dsoscaxis_c);  

if( lspg_nextshot.dsoscaxis != NULL)  

    free( lspg_nextshot.dsoscaxis);  

lspg_nextshot.dsoscaxis = strdup( PQgetvalue( pgr, 0,  

    dsoscaxis_c));  

lspg_nextshot.dsoscaxis2_isnull = PQgetisnull(  

    pgr, 0, dsoscaxis2_c);  

if( lspg_nextshot.dsoscaxis2 != NULL)  

    free( lspg_nextshot.dsoscaxis2);  

lspg_nextshot.dsoscaxis2 = strdup( PQgetvalue( pgr, 0,  

    dsoscaxis2_c));  

lspg_nextshot.sfn_isnull = PQgetisnull(pgr, 0, sfn_c);  

if( lspg_nextshot.sfn != NULL)  

    free( lspg_nextshot.sfn);  

lspg_nextshot.sfn = strdup( PQgetvalue( pgr, 0, sfn_c));  

lspg_nextshot.stype_isnull = PQgetisnull( pgr, 0,  

    stype_c);  

if( lspg_nextshot.stype != NULL)  

    free( lspg_nextshot.stype);  

lspg_nextshot.stype = strdup( PQgetvalue( pgr, 0, stype_c))  

;  

lspg_nextshot.stype2_isnull = PQgetisnull( pgr, 0,  

    stype2_c);  

if( lspg_nextshot.stype2 != NULL)  

    free( lspg_nextshot.stype2);  

lspg_nextshot.stype2 = strdup( PQgetvalue( pgr, 0,  

    stype2_c));  

//  

// Probably shouldn't try to convert null number values  

//  

lspg_nextshot.dsowidth_isnull = PQgetisnull( pgr,

```

```

        0, dsowidth_c);
if( lsgp_nextshot.dsowidth_isnull == 0)
    lsgp_nextshot.dsowidth = atof( PQgetvalue( pgr,0,
        dsowidth_c));

lsgp_nextshot.dsexp_isnull = PQgetisnull( pgr, 0,
    dsexp_c);
if( lsgp_nextshot.dsexp_isnull == 0)
    lsgp_nextshot.dsexp      = atof( PQgetvalue( pgr,0, dsexp_c
        ));

lsgp_nextshot.sstart_isnull = PQgetisnull( pgr, 0,
    sstart_c);
if( lsgp_nextshot.sstart_isnull == 0)
    lsgp_nextshot.sstart     = atof( PQgetvalue( pgr,0, sstart_c));

lsgp_nextshot.dsphi_isnull = PQgetisnull( pgr, 0,
    dsphi_c);
if( lsgp_nextshot.dsphi_isnull == 0)
    lsgp_nextshot.dsphi      = atof( PQgetvalue( pgr,0, dsphi_c
        ));

lsgp_nextshot.dsomega_isnull = PQgetisnull( pgr, 0
    , dsomega_c);
if( lsgp_nextshot.dsomega_isnull == 0)
    lsgp_nextshot.dsomega   = atof( PQgetvalue( pgr,0, dsomega_c));

lsgp_nextshot.dskappa_isnull = PQgetisnull( pgr, 0
    , dskappa_c);
if( lsgp_nextshot.dskappa_isnull == 0)
    lsgp_nextshot.dskappa   = atof( PQgetvalue( pgr,0, dskappa_c));

lsgp_nextshot.dsdist_isnull = PQgetisnull( pgr, 0
    , dsdist_c);
if( lsgp_nextshot.dsdist_isnull == 0)
    lsgp_nextshot.dsdist    = atof( PQgetvalue( pgr,0, dsdist_c));

lsgp_nextshot.dsnrq_isnull = PQgetisnull( pgr, 0,
    dsnrq_c);
if( lsgp_nextshot.dsnrq_isnull == 0)
    lsgp_nextshot.dsnrq     = atof( PQgetvalue( pgr,0, dsnrq_c
        ));

lsgp_nextshot.cx_isnull = PQgetisnull( pgr, 0, cx_c);
if( lsgp_nextshot.cx_isnull == 0)
    lsgp_nextshot.cx        = atof( PQgetvalue( pgr,0, cx_c));

lsgp_nextshot.cy_isnull = PQgetisnull( pgr, 0, cy_c);
if( lsgp_nextshot.cy_isnull == 0)
    lsgp_nextshot.cy        = atof( PQgetvalue( pgr,0, cy_c));

lsgp_nextshot.ax_isnull = PQgetisnull( pgr, 0, ax_c);
if( lsgp_nextshot.ax_isnull == 0)
    lsgp_nextshot.ax        = atof( PQgetvalue( pgr,0, ax_c));

lsgp_nextshot.ay_isnull = PQgetisnull( pgr, 0, ay_c);
if( lsgp_nextshot.ay_isnull == 0)
    lsgp_nextshot.ay        = atof( PQgetvalue( pgr,0, ay_c));

lsgp_nextshot.az_isnull = PQgetisnull( pgr, 0, az_c);
if( lsgp_nextshot.az_isnull == 0)
    lsgp_nextshot.az        = atof( PQgetvalue( pgr,0, az_c));

lsgp_nextshot.active_isnull = PQgetisnull( pgr, 0,
    active_c);
if( lsgp_nextshot.active_isnull == 0)
    lsgp_nextshot.active   = atoi( PQgetvalue( pgr, 0,
        active_c));

lsgp_nextshot.sindex_isnull = PQgetisnull( pgr, 0,
    sindex_c);
if( lsgp_nextshot.sindex_isnull == 0)
    lsgp_nextshot.sindex   = atoi( PQgetvalue( pgr, 0,
        sindex_c));

lsgp_nextshot.dshpid_isnull = PQgetisnull( pgr, 0,
    dshpid_c);
if( lsgp_nextshot.dshpid_isnull == 0)
    lsgp_nextshot.dshpid   = atoi( PQgetvalue( pgr, 0,
        dshpid_c));

lsgp_nextshot.skey_isnull = PQgetisnull( pgr, 0,
    skey_c);

```

```

if( lsgc_nextshot.skey_isnull == 0)
lsgc_nextshot.skey    = atol( PQgetvalue( pgr, 0, skey_c))
;

lsgc_nextshot.dsowidth2_isnull = PQgetisnull(
    pgr, 0, dsowidth2_c);
if( lsgc_nextshot.dsowidth2_isnull == 0)
lsgc_nextshot.dsowidth2 = atof( PQgetvalue( pgr,0,
    dsowidth2_c));

lsgc_nextshot.dsexp2_isnull = PQgetisnull( pgr, 0,
    dsexp2_c);
if( lsgc_nextshot.dsexp2_isnull == 0)
lsgc_nextshot.dsexp2    = atof( PQgetvalue( pgr,0,
    dsexp2_c));

lsgc_nextshot.sstart2_isnull = PQgetisnull( pgr, 0
    , sstart2_c);
if( lsgc_nextshot.sstart2_isnull == 0)
lsgc_nextshot.sstart2   = atof( PQgetvalue( pgr,0,
    sstart2_c));

lsgc_nextshot.dsphi2_isnull = PQgetisnull( pgr, 0,
    dsphi2_c);
if( lsgc_nextshot.dsphi2_isnull == 0)
lsgc_nextshot.dsphi2    = atof( PQgetvalue( pgr,0,
    dsphi2_c));

lsgc_nextshot.dsomega2_isnull = PQgetisnull( pgr,
    0, dsomega2_c);
if( lsgc_nextshot.dsomega2_isnull == 0)
lsgc_nextshot.dsomega2  = atof( PQgetvalue( pgr,0,
    dsomega2_c));

lsgc_nextshot.dskappa2_isnull = PQgetisnull( pgr,
    0, dskappa2_c);
if( lsgc_nextshot.dskappa2_isnull == 0)
lsgc_nextshot.dskappa2  = atof( PQgetvalue( pgr,0,
    dskappa2_c));

lsgc_nextshot.dsdist2_isnull = PQgetisnull( pgr, 0
    , dsdist2_c);
if( lsgc_nextshot.dsdist2_isnull == 0)
lsgc_nextshot.dsdist2   = atof( PQgetvalue( pgr,0,
    dsdist2_c));

lsgc_nextshot.dsnrq2_isnull = PQgetisnull( pgr, 0,
    dsnrq2_c);
if( lsgc_nextshot.dsnrq2_isnull == 0)
lsgc_nextshot.dsnrq2    = atof( PQgetvalue( pgr,0,
    dsnrq2_c));

lsgc_nextshot.cx2_isnull = PQgetisnull( pgr, 0, cx2_c)
;
if( lsgc_nextshot.cx2_isnull == 0)
lsgc_nextshot.cx2      = atof( PQgetvalue( pgr,0, cx2_c));

lsgc_nextshot.cy2_isnull = PQgetisnull( pgr, 0, cy2_c)
;
if( lsgc_nextshot.cy2_isnull == 0)
lsgc_nextshot.cy2      = atof( PQgetvalue( pgr,0, cy2_c));

lsgc_nextshot.ax2_isnull = PQgetisnull( pgr, 0, ax2_c)
;
if( lsgc_nextshot.ax2_isnull == 0)
lsgc_nextshot.ax2      = atof( PQgetvalue( pgr,0, ax2_c));

lsgc_nextshot.ay2_isnull = PQgetisnull( pgr, 0, ay2_c)
;
if( lsgc_nextshot.ay2_isnull == 0)
lsgc_nextshot.ay2      = atof( PQgetvalue( pgr,0, ay2_c));

lsgc_nextshot.az2_isnull = PQgetisnull( pgr, 0, az2_c)
;
if( lsgc_nextshot.az2_isnull == 0)
lsgc_nextshot.az2      = atof( PQgetvalue( pgr,0, az2_c));

lsgc_nextshot.active2_isnull = PQgetisnull( pgr, 0
    , active2_c);
if( lsgc_nextshot.active2_isnull == 0)
lsgc_nextshot.active2  = atoi( PQgetvalue( pgr, 0,
    active2_c));

lsgc_nextshot.sindex2_isnull = PQgetisnull( pgr, 0
    , sindex2_c);
if( lsgc_nextshot.sindex2_isnull == 0)
lsgc_nextshot.sindex2  = atoi( PQgetvalue( pgr, 0,
    sindex2_c));

```

```

    sindex2_c));
lspg_nextshot.new_value_ready = 1;
pthread_cond_signal( &(lspg_nextshot.cond));
pthread_mutex_unlock( &(lspg_nextshot.mutex));
}

```

#### 7.4.4.46 void lspg\_nextshot\_done( )

Called when the next shot query has been processed.

Definition at line 933 of file Lspg.c.

```

{
pthread_mutex_unlock( &(lspg_nextshot.mutex));
}
```

#### 7.4.4.47 void lspg\_nextshot\_init( )

Initialize the nextshot variable, mutex, and condition.

Definition at line 907 of file Lspg.c.

```

{
memset( &lspg_nextshot, 0, sizeof( lspg_nextshot));
pthread_mutex_init( &(lspg_nextshot.mutex), NULL);
pthread_cond_init( &(lspg_nextshot.cond), NULL);
}
```

#### 7.4.4.48 void lspg\_nextshot\_wait( )

Wait for the next shot query to get processed.

Definition at line 925 of file Lspg.c.

```

{
pthread_mutex_lock( &(lspg_nextshot.mutex));
while( lspg_nextshot.new_value_ready == 0)
    pthread_cond_wait( &(lspg_nextshot.cond), &(lspg_nextshot
        .mutex));
}
```

#### 7.4.4.49 PQnoticeProcessor lspg\_notice\_processor( void \* arg, const char \* msg )

Definition at line 1647 of file Lspg.c.

```

{
lslogging_log_message( "lspg: %s", msg);
return NULL;
}
```

#### 7.4.4.50 void lspg\_pg\_connect( )

Connect to the pg server.

Definition at line 1654 of file Lspg.c.

```

    {
int err;

if( q == NULL)
    ls_pg_state = LS_PG_STATE_INIT;

switch( ls_pg_state) {
case LS_PG_STATE_INIT:

    if( lspg_time_sent.tv_sec != 0) {
        //
        // Reality check: if it's less than about 10 seconds since the last failed
        attempt
        // the just chill.
        //
        gettimeofday( &now, NULL);
        if( now.tv_sec - lspg_time_sent.tv_sec < 10) {
            return;
        }
    }

    q = PQconnectStart( "dbname=ls user=lsuser hostaddr=10.1.0.3");
    if( q == NULL) {
        lslogging_log_message( "Out of memory
        (lspg_pg_connect)");
        exit( -1);
    }

    err = PQstatus( q);
    if( err == CONNECTION_BAD) {
        lslogging_log_message( "Trouble connecting to
        database");

        gettimeofday( &lspg_time_sent, NULL);
        return;
    }
    err = PQsetnonblocking( q, 1);
    if( err != 0) {
        lslogging_log_message( "Odd, could not set database
        connection to nonblocking");
    }

    ls_pg_state = LS_PG_STATE_INIT_POLL;
    lspg_connectPoll_response = PGRES_POLLING_WRITING;
    //
    // set up the connection for poll
    //
    lspgfd.fd = PQsocket( q);
    break;

case LS_PG_STATE_INIT_POLL:
    if( lspg_connectPoll_response ==
        PGRES_POLLING_FAILED) {
        PQfinish( q);
        q = NULL;
        ls_pg_state = LS_PG_STATE_INIT;
    } else if( lspg_connectPoll_response ==
        PGRES_POLLING_OK) {
        PQsetNoticeProcessor( q, (PQnoticeProcessor)lspg_notice_processor
        , NULL);
        lspg_query_push( NULL, "select pmac.md2_init()");
        ls_pg_state = LS_PG_STATE_IDLE;
    }
    break;

case LS_PG_STATE_RESET:
    err = PQresetStart( q);
    if( err == 0) {
        PQfinish( q);
        q = NULL;
        ls_pg_state = LS_PG_STATE_INIT;
    } else {
        ls_pg_state = LS_PG_STATE_RESET_POLL;
        lspg_resetPoll_response = PGRES_POLLING_WRITING;
    }
    break;

case LS_PG_STATE_RESET_POLL:
    if( lspg_resetPoll_response == PGRES_POLLING_FAILED)
    {
        PQfinish( q);
        q = NULL;
        ls_pg_state = LS_PG_STATE_INIT;
    } else if( lspg_resetPoll_response ==
        PGRES_POLLING_OK) {
        lspg_query_push( NULL, "select pmac.md2_init()");
        ls_pg_state = LS_PG_STATE_IDLE;
    }
}

```

```

        }
        break;
    }
}
```

#### 7.4.4.51 void lsgpg\_pg\_service ( struct pollfd \* evt )

I/O control to/from the postgresql server.

##### Parameters

in	evt	The pollfd object that we are responding to
----	-----	---

Definition at line 1544 of file lsgpg.c.

```

{
// Currently just used to check for notifies
// Other socket communication is done synchronously
//

if( evt->revents & POLLIN) {
    int err;

    if( ls_pg_state == LS_PG_STATE_INIT_POLL) {
        lsgpg_connectPoll_response = PQconnectPoll( q);
        if( lsgpg_connectPoll_response ==
            PGRES_POLLING_FAILED) {
            ls_pg_state = LS_PG_STATE_RESET;
        }
        return;
    }

    if( ls_pg_state == LS_PG_STATE_RESET_POLL)
    {
        lsgpg_resetPoll_response = PQresetPoll( q);
        if( lsgpg_resetPoll_response ==
            PGRES_POLLING_FAILED) {
            ls_pg_state = LS_PG_STATE_RESET;
        }
        return;
    }

    //
    // if in IDLE or RECV we need to call consumeInput first
    //
    if( ls_pg_state == LS_PG_STATE_IDLE) {
        err = PQconsumeInput( q);
        if( err != 1) {
            lslogging_log_message( "consume input failed: %s",
                PQerrorMessage( q));
            ls_pg_state = LS_PG_STATE_RESET;
            return;
        }
    }

    if( ls_pg_state == LS_PG_STATE_RECV) {
        lsgpg_receive();
    }

    //
    // Check for notifies regardless of our state
    // Push as many requests as we have notifies.
    //
    {
        PGnotify *pgn;

        while( 1) {
            pgn = PQnotifies( q);
            if( pgn == NULL)
                break;

            lslogging_log_message( "lsgpg_pg_service: notify
recieved %s", pgn->relname);

            if( strstr( pgn->relname, "_pmac") != NULL) {
                lsgpg_query_push( lsgpg_cmd_cb, "EXECUTE
md2_queue_next");
            }
        }
    }
}
}
```

```

    } else if( strstr( pgn->relname, "_diff") != NULL || strstr( pgn->
    relname, "_run") != NULL) {
        lsgp_query_push( lsgp_nextaction_cb,
    "EXECUTE nextaction");
    } else if( strstr( pgn->relname, "_sample") != NULL) {
        lsgp_getcurrentsampleid_call();
    } else if( strstr( pgn->relname, "_kvs") != NULL) {
        lsgp_query_push( lsgp_allkvs_cb, "
    EXECUTE getkvs");
    } else if( strstr( pgn->relname, "_mess") != NULL) {
        lsgp_query_push( lsgp_nexterrors_cb,
    "EXECUTE nexterrors");
    }
    PQfreemem( pgn);
}
}

if( evt->revents & POLLOUT) {

if( ls_pg_state == LS_PG_STATE_INIT_POLL) {
    lsgp_connectPoll_response = PQconnectPoll( q);
    if( lsgp_connectPoll_response ==
PGRES_POLLING_FAILED) {
        ls_pg_state = LS_PG_STATE_RESET;
    }
    return;
}

if( ls_pg_state == LS_PG_STATE_RESET_POLL)
{
    lsgp_resetPoll_response = PQresetPoll( q);
    if( lsgp_resetPoll_response ==
PGRES_POLLING_FAILED) {
        ls_pg_state = LS_PG_STATE_RESET;
    }
    return;
}

if( ls_pg_state == LS_PG_STATE_SEND) {
    lsgp_send_next_query();
}

if( ls_pg_state == LS_PG_STATE_SEND_FLUSH)
{
    lsgp_flush();
}
}

```

**7.4.4.52 void lsgp\_preset\_changed\_cb ( char \* event )**

Definition at line 1864 of file lsgc.c.

```
static char base[] = "Preset Changed ";
char *pn;
lsredis_obj_t *p;
char *v;

pn = strstr( event, base);
if( pn == NULL) {
    lslogging_log_message( "lspg_preset_changed_cb: Could
        not parse '%s'", event);
    return;
}
pn += strlen( base);

p = lsredis_get_obj( "%s", pn);
if( p == NULL) {
    lslogging_log_message( "lspg_preset_changed_cb: Could
        not find variable '%s'", pn);
    return;
}
v = lsredis_getstr( p);
if( v == NULL || v[0] == 0) {
    lslogging_log_message( "lspg_preset_chanted_cb: Value
        for preset %s is %s", pn, v==NULL ? "NULL" : "Empty");
    return;
}
lspg_query_push( NULL, "EXECUTE kvupdate('{%,%s}'::text[]")";
```

```
    pn, v);
}
```

#### 7.4.4.53 `lspg_query_queue_t* lspg_query_next( )`

Return the next item in the postgresql queue.

If there is an item left in the queue then it is returned. Otherwise, NULL is returned.

Definition at line 74 of file `lspg.c`.

```
{
lspg_query_queue_t *rtn;

pthread_mutex_lock( &lspg_query_queue_mutex);

if( lspg_query_queue_off == lspg_query_queue_on
)
// Queue is empty
rtn = NULL;
else {
    rtn = &(lspg_query_queue[(lspg_query_queue_off
    ++) % LS_PG_QUERY_QUEUE_LENGTH]);
    pthread_cond_signal( &lspg_query_cond);
}
pthread_mutex_unlock( &lspg_query_mutex);

return rtn;
}
```

#### 7.4.4.54 `void lspg_query_push( void()(lspg_query_queue_t *, PGresult *) cb, char * fmt, ... )`

Place a query on the queue.

##### Parameters

in	<i>cb</i>	Our callback function that deals with the response
in	<i>fmt</i>	Printf style function to generate the query

Definition at line 127 of file `lspg.c`.

```
{
int idx;
va_list arg_ptr;

pthread_mutex_lock( &lspg_query_mutex);

//
// Pause the thread while we service the queue
//
while( (lspg_query_queue_on + 1) %
    LS_PG_QUERY_QUEUE_LENGTH == lspg_query_queue_off %
    LS_PG_QUERY_QUEUE_LENGTH) {
    pthread_cond_wait( &lspg_query_cond, &lspg_query_mutex
    );
}

idx = lspg_query_queue_on % LS_PG_QUERY_QUEUE_LENGTH
;

va_start( arg_ptr, fmt);
vsnprintf( lspg_query_queue[idx].qs,
    LS_PG_QUERY_STRING_LENGTH-1, fmt, arg_ptr);
va_end( arg_ptr);

lspg_query_queue[idx].qs[LS_PG_QUERY_STRING_LENGTH
    - 1] = 0;
lspg_query_queue[idx].onResponse = cb;
lspg_query_queue_on++;

pthread_kill( lspg_thread, SIGUSR1);
pthread_mutex_unlock( &lspg_query_mutex);
};
```

#### 7.4.4.55 void lspg\_query\_reply\_next( )

Remove the oldest item in the queue.

this is called only when there is nothing else to service the reply: this pop does not return anything. We use the ...reply\_peek function to return the next item in the reply queue

Definition at line 98 of file lspg.c.

```
{
pthread_mutex_lock( &lspg_queue_mutex );
if( lspg_query_queue_reply != lspg_query_queue_on )
    lspg_query_queue_reply++;
pthread_mutex_unlock( &lspg_queue_mutex );
}
```

#### 7.4.4.56 lspg\_query\_queue\_t\* lspg\_query\_reply\_peek( )

Return the next item in the reply queue but don't pop it since we may need it more than once.

Call `lspg_query_reply_next()` when done.

Definition at line 111 of file lspg.c.

```
{
lspg_query_queue_t *rtn;
pthread_mutex_lock( &lspg_queue_mutex );
if( lspg_query_queue_reply == lspg_query_queue_on )
    rtn = NULL;
else
    rtn = &(lspg_query_queue[(lspg_query_queue_reply
        ) % LS_PG_QUERY_QUEUE_LENGTH]);
pthread_mutex_unlock( &lspg_queue_mutex );
return rtn;
}
```

#### 7.4.4.57 void lspg\_quitting\_cb( char \* event )

Prepare to exit the program in a couple of seconds.

Definition at line 1973 of file lspg.c.

```
{
lspg_query_push( NULL, "SELECT px.dropairrights()");
```

#### 7.4.4.58 void lspg\_receive( )

Receive a result of a query.

Definition at line 1461 of file lspg.c.

```
{
PGresult *pgr;
lspg_query_queue_t *qqp;
int err;
err = PQconsumeInput( q );
if( err != 1 ) {
    lslogging_log_message( "consume input failed: %s",
    "
```

```

PQerrorMessage( q );
ls_pg_state = LS_PG_STATE_RESET;
return;
}

// We must call PQgetResult until it returns NULL before sending the next
// query
// This implies that only one query can ever be active at a time and our
// queue
// management should be simple
//
// We should be in the LS_PG_STATE_RECV here
//

while( !PQisBusy( q ) ) {
    pgr = PQgetResult( q );
    if( pgr == NULL ) {
        lsgpq_query_reply_next();
        //
        // we are now done reading the response from the database
        //
        ls_pg_state = LS_PG_STATE_IDLE;
        break;
    } else {
        ExecStatusType es;

        qqp = lsgpq_query_reply_peek();
        es = PQresultStatus( pgr );

        if( es != PGRES_COMMAND_OK && es != PGRES_TUPLES_OK ) {
            char *emess;
            emess = PQresultErrorMessage( pgr );
            if( emess != NULL && emess[0] != 0 ) {
                lsllogging_log_message( "Error from query '%s':\n"
                "%s", qqp->qs, emess );
            }
        } else {
            //
            // Deal with the response
            //
            // If the response is likely to take awhile we should probably
            // add a new state and put something in the main loop to run the
            onResponse
            // routine in the main loop. For now, though, we only expect very
            brief onResponse routines
            //
            if( qqp != NULL && qqp->onResponse != NULL )
                qqp->onResponse( qqp, pgr );
        }
        PQclear( pgr );
    }
}

```

#### 7.4.4.59 void lsgp\_run( )

Start 'er runnin'.

Definition at line 1998 of file lsgc.c.

```

pthread_create( &lspg_thread, NULL, lspg_worker, NULL);
lsevents_add_listener( "^(appy|appz|capy|capz|scint) In
    Position$", lspg_check_preset_in_position_cb);
lsevents_add_listener( "^(appy|appz|capy|capz|scint)
    Moving$",      lspg_unset_current_preset_moving_cb
);
lsevents_add_listener( "Preset Changed (.+)",
    lspg_preset_changed_cb);
lsevents_add_listener( "^Sample(Detected|Absent)$",
    lspg_sample_detector_cb);
lsevents_add_listener( "^Timer Update KVs$",
    lspg_update_kvs_cb);
lsevents_add_listener( "^cam.zoom In Position$",
    lspg_set_scale_cb);
lstimter_set_timer( "Timer Update KVs", -1, 0, 5000000000
;
// Make sure we own the airrights
//

```

```

    lspg_demandairrights_all();
}

```

#### 7.4.4.60 void lspg\_sample\_detector\_cb ( char \* event )

log magnet state

Definition at line 1961 of file lspg.c.

```

{
    int present;
    if( strcmp( event, "SampleDetected") == 0)
        present = 1;
    else
        present = 0;

    lspg_query_push( NULL, "SELECT px.logmagnetstate(%s)", present
                    ? "TRUE" : "FALSE");
}

```

#### 7.4.4.61 void lspg\_send\_next\_query ( )

send the next queued query to the DB server

Definition at line 1414 of file lspg.c.

```

{
// Normally we should be in the "send" state
// but we can also send if we are servicing
// a reply
//

lspg_query_queue_t *qqp;
int err;

qqp = lspg_query_next();
if( qqp == NULL) {
    //
// A send without a query? Should never happen.
// But at least we shouldn't segfault if it does.
//
return;
}

if( qqp->qs[0] == 0) {
    //
// Do we really have to check this case?
// It would only come up if we stupidly pushed an empty query string
// or ran off the end of the queue
//
lslogging_log_message( "Popped empty query string.
    Probably bad things are going on.");

lspg_query_reply_next();
ls_pg_state = LS_PG_STATE_IDLE;
} else {
    err = PQsendQuery( q, qqp->qs);
    if( err == 0) {
        lslogging_log_message( "query failed: %s\n",
        PQerrorMessage( q));

        //
// Don't wait for a reply, just reset the connection
//
        lspg_query_reply_next();
        ls_pg_state = LS_PG_STATE_RESET;
    } else {
        ls_pg_state = LS_PG_STATE_SEND_FLUSH;
    }
}

```

---

**7.4.4.62 void lsgp\_seq\_run\_prep\_all ( long long *skey*, double *kappa*, double *phi*, double *cx*, double *cy*, double *ax*, double *ay*, double *az* )**

Convinence function to call seq run prep.

#### Parameters

in	<i>skey</i>	px.shots key for this image
in	<i>kappa</i>	current kappa position
in	<i>phi</i>	current phi position
in	<i>cx</i>	current center table x
in	<i>cy</i>	current center table y
in	<i>ax</i>	current alignment table x
in	<i>ay</i>	current alignment table y
in	<i>az</i>	current alignment table z

Definition at line 1186 of file lsgp.c.

```
{
lsgp_seq_run_prep_call( skey, kappa, phi, cx,
                      cy, ax, ay, az);
lsgp_seq_run_prep_wait();
lsgp_seq_run_prep_done();
}
```

---

**7.4.4.63 void lsgp\_seq\_run\_prep\_call ( long long *skey*, double *kappa*, double *phi*, double *cx*, double *cy*, double *ax*, double *ay*, double *az* )**

queue up the seq\_run\_prep query

#### Parameters

in	<i>skey</i>	px.shots key for this image
in	<i>kappa</i>	current kappa position
in	<i>phi</i>	current phi position
in	<i>cx</i>	current center table x
in	<i>cy</i>	current center table y
in	<i>ax</i>	current alignment table x
in	<i>ay</i>	current alignment table y
in	<i>az</i>	current alignment table z

Definition at line 1152 of file lsgp.c.

```
{
pthread_mutex_lock( &(lsgp_seq_run_prep.mutex));
lsgp_seq_run_prep.new_value_ready = 0;
pthread_mutex_unlock( &(lsgp_seq_run_prep.mutex));

lsgp_query_push( lsgp_seq_run_prep_cb,
                  "SELECT px.seq_run_prep( %lld, %.3f, %.3f, %.3f, %.3f, %.3f, %.3f)",
                  skey, kappa, phi, cx, cy, ax, ay, az);
}
```

---

**7.4.4.64 void lsgp\_seq\_run\_prep\_cb ( lsgp\_query\_queue\_t \* *qqp*, PGresult \* *pgr* )**

Callback for the seq\_run\_prep query.

#### Parameters

in	<i>qqp</i>	The query item that generated this callback
in	<i>pgr</i>	The result of the query

Definition at line 1140 of file lspg.c.

```
{
pthread_mutex_lock( &(lspg_seq_run_prep.mutex));
lspg_seq_run_prep.new_value_ready = 1;
pthread_cond_signal( &(lspg_seq_run_prep.cond));
pthread_mutex_unlock( &(lspg_seq_run_prep.mutex));
}
```

#### 7.4.4.65 void lspg\_seq\_run\_prep\_done( )

Indicate we are done waiting.

Definition at line 1180 of file lspg.c.

```
{
pthread_mutex_unlock( &(lspg_seq_run_prep.mutex));
}
```

#### 7.4.4.66 void lspg\_seq\_run\_prep\_init( )

Initialize the data collection object.

Definition at line 1132 of file lspg.c.

```
{
lspg_seq_run_prep.new_value_ready = 0;
pthread_mutex_init( &(lspg_seq_run_prep.mutex), NULL);
pthread_cond_init( &(lspg_seq_run_prep.cond), NULL);
}
```

#### 7.4.4.67 void lspg\_seq\_run\_prep\_wait( )

Wait for seq run prep query to return.

Definition at line 1172 of file lspg.c.

```
{
pthread_mutex_lock( &(lspg_seq_run_prep.mutex));
while( lspg_seq_run_prep.new_value_ready == 0
      )
pthread_cond_wait( &(lspg_seq_run_prep.cond), &
                  lspg_seq_run_prep.mutex));
}
```

#### 7.4.4.68 void lspg\_set\_scale\_cb( char \* event )

Fix up xscale and yscale when zoom changes.

Definition at line 1934 of file lspg.c.

```
{
int mag;
lsredis_obj_t *px, *py;
char *sx, *sy;

//
// There is already a call back to set the redis variables xScale and yScale
// we just need to set the KV's
//

mag = lspmacc_getPosition( zoom);

px = lsredis_get_obj( "cam.zoom.%d.ScaleX", mag);
sx = lsredis_getstr( px);
```

```

py = lsredis_get_obj( "cam.zoom.%d.ScaleY", mag);
sy = lsredis_getstr( py);

lspg_query_push( NULL, "EXECUTE kvupdate(
    '{cam.xScale,%s,cam.yScale,%s}')", sx, sy);
free( sx);
free( sy);
}

```

#### 7.4.4.69 void lspg\_sig\_service ( struct pollfd \* evt )

Service a signal Signals here are treated as file descriptors and fits into our poll scheme.

##### Parameters

in	evt	The pollfd object that triggered this call
----	-----	--

Definition at line 1522 of file lspg.c.

```

{
struct signalfd_siginfo fdsi;

// Really, we don't care about the signal,
// it's just used to drop out of the poll
// function when there is something for us
// to do that didn't invoke something coming
// from our postgresql server.
//
// This is accomplished by the query_push function
// to notify us that a new query is ready.
//

read( evt->fd, &fdsi, sizeof( struct signalfd_siginfo));
}


```

#### 7.4.4.70 int lspg\_starttransfer\_all ( int \* err, unsigned int nextsample, int sampledetected, double ax, double ay, double az, double horz, double vert, double esttime )

Definition at line 409 of file lspg.c.

```

int rtn;

lspg_starttransfer_call( nextsample, sampledetected,
    ax, ay, az, horz, vert, esttime);
lspg_starttransfer_wait();
if( lspg_starttransfer.no_rows_returned ||
    lspg_starttransfer.starttransfer != 1) {
    *err = 1;
} else {
    *err = 0;
    rtn = lspg_starttransfer.starttransfer;
}
lspg_starttransfer_done();

return rtn;
}


```

#### 7.4.4.71 void lspg\_starttransfer\_call ( unsigned int nextsample, int sample\_detected, double ax, double ay, double az, double horz, double vert, double esttime )

Definition at line 389 of file lspg.c.

```

    {
pthread_mutex_lock( &(lspg_starttransfer.mutex));
lspg_starttransfer.new_value_ready = 0;
pthread_mutex_unlock( &(lspg_starttransfer.mutex));

lspg_query_push( lspg_starttransfer_cb,
    "SELECT px.starttransfer( %d, %s, %.3f, %.3f, %.3f, %.3f, %.3f )",
    nextsample, sample_detected ? "True" : "False
    ", ax, ay, az, horz, vert, esttime);
}

```

#### 7.4.4.72 void lspg\_starttransfer\_cb ( lspg\_query\_queue\_t \* qqp, PGresult \* pgr )

##### Parameters

in	qqp	Our nextsample query
in	pgr	result of the query

Definition at line 367 of file lspg.c.

```

    {
pthread_mutex_lock( &(lspg_starttransfer.mutex));

lspg_starttransfer.new_value_ready = 1;
if( PQntuples( pgr ) <=0 ) {
    lspg_starttransfer.no_rows_returned = 1;
    lspg_starttransfer.starttransfer = 0;
} else {
    lspg_starttransfer.no_rows_returned = 0;
    lslogging_log_message( "lspg_starttransfer_cb:
        received '%s' from strattransfer query", PQgetvalue( pgr,0,0));
    if( PQgetisnull( pgr, 0, 0 ) || strcmp( PQgetvalue( pgr,0,0 ), "1" ) != 0
        lspg_starttransfer.starttransfer = 0;
    else
        lspg_starttransfer.starttransfer = 1;
}
pthread_cond_signal( &(lspg_starttransfer.cond));
pthread_mutex_unlock( &(lspg_starttransfer.mutex));
}
```

#### 7.4.4.73 void lspg\_starttransfer\_done ( )

Definition at line 404 of file lspg.c.

```

    {
pthread_mutex_unlock( &(lspg_starttransfer.mutex));
}
```

#### 7.4.4.74 void lspg\_starttransfer\_init ( )

Definition at line 361 of file lspg.c.

```

    {
lspg_starttransfer.new_value_ready = 0;
pthread_mutex_init( &lspg_starttransfer.mutex, NULL);
pthread_cond_init( &lspg_starttransfer.cond, NULL);
}
```

#### 7.4.4.75 void lspg\_starttransfer\_wait ( )

Definition at line 398 of file lspg.c.

```

    {
pthread_mutex_lock( &(lspg_starttransfer.mutex));
while( lspg_starttransfer.new_value_ready ==
      0)
pthread_cond_wait( &(lspg_starttransfer.cond), &
                  lspg_starttransfer.mutex));
}

```

#### 7.4.4.76 void lspg\_unset\_current\_preset\_moving\_cb ( char \* event )

Definition at line 1911 of file lspg.c.

```

{
lspmac_motor_t *mp;
char cp[64];
int i;

for( i=0; i<strlen( event); i++) {
    cp[i] = 0;
    if( event[i] == ' ')
        break;
    cp[i] = event[i];
}

mp = lspmac_find_motor_by_name( cp);
if( mp == NULL) {
    lslogging_log_message( "
    lspg_unset_current_reset_moving_cb: Could not find motor '%s'", cp);
    return;
}
lspg_query_push( NULL, "EXECUTE kvupdate(
    '{%s.currentPreset,-1}'", cp);
}

```

#### 7.4.4.77 void lspg\_update\_kvs\_cb ( char \* event )

Perhaps update the px.kvs table in postgresql Should be triggered by a timer event.

Definition at line 304 of file lspg.c.

```

{
static char s[LS_PG_QUERY_STRING_LENGTH - 64], *fmt;
int i, need_comma, n;
lspmac_motor_t *mp;
int updateme;
double new_value;

s[0] = 0;
need_comma = 0;

for( i=0; i<lspmac_nmotors; i++ ) {
    mp = &(lspmac_motors[i]);
    pthread_mutex_lock( &mp->mutex);
    if( fabs(mp->reported_pg_position - mp->position
         ) >= lsredis_getd(mp->update_resolution)) {
        new_value = mp->position;
        mp->reported_pg_position = mp->position;
        fmt = lsredis_getstr( mp->redis_fmt);           // borrow the redis format
        updateme = 1;
    } else {
        updateme = 0;
    }
    pthread_mutex_unlock( &mp->mutex);
    if( !updateme)
        continue;

    n = strlen( s);
    snprintf( &(s[n]), sizeof(s)-n-1, "%s%s.position,", need_comma++ ? "," : ""
              , mp->name);

    n = strlen( s);
    snprintf( &(s[n]), sizeof(s)-n-1, fmt, new_value);

    //
    // And again for the original remote interface
    // We'll be able to remove this, someday
}

```

```

//  

n = strlen( s );  

snprintf( &(s[n]), sizeof(s)-n-1, ",%s,", mp->name );  

  

n = strlen( s );  

snprintf( &(s[n]), sizeof(s)-n-1, fmt, new_value );  

free( fmt );
  

n = strlen( s );
if( n >= sizeof(s) - 64 ) {
    lsgp_query_push( NULL, "EXECUTE kvupdate('{%s}')", s );
    s[0] = 0;
    need_comma = 0;
}
}  

  

if( strlen(s) ) {
    lsgp_query_push( NULL, "EXECUTE kvupdate('{%s}')", s );
}
}

```

#### 7.4.4.78 void lsgp\_wait\_for\_detector\_all( )

Combined call to wait for the detector.

Definition at line 996 of file lspg.c.

```

{
lsgp_wait_for_detector_call();
lsgp_wait_for_detector_wait();
lsgp_wait_for_detector_done();
}

```

#### 7.4.4.79 void lsgp\_wait\_for\_detector\_call( )

initiate the wait for detector query

Definition at line 970 of file lspg.c.

```

{
pthread_mutex_lock( &(lsgp_wait_for_detector.mutex
));
lsgp_wait_for_detector.new_value_ready =
0;
pthread_mutex_unlock( &(lsgp_wait_for_detector.mutex
));
  

lsgp_query_push( lsgp_wait_for_detector_cb
, "SELECT px.lock_detector_test_block()");
}

```

#### 7.4.4.80 void lsgp\_wait\_for\_detector\_cb( lsgp\_query\_queue\_t \* qqp, PGresult \* pgr )

Callback for the wait for detector query.

Definition at line 961 of file lspg.c.

```

{
pthread_mutex_lock( &(lsgp_wait_for_detector.mutex
));
lsgp_wait_for_detector.new_value_ready =
1;
pthread_cond_signal( &(lsgp_wait_for_detector.cond
));
pthread_mutex_unlock( &(lsgp_wait_for_detector.mutex
));
}

```

#### 7.4.4.81 void lsgp\_wait\_for\_detector\_done( )

Done waiting for the detector.

Definition at line 989 of file lsgp.c.

```
{
    pthread_mutex_unlock( &(lsgp_wait_for_detector.mutex
        ));
}
```

#### 7.4.4.82 void lsgp\_wait\_for\_detector\_init( )

initialize the detector timing object

Definition at line 953 of file lsgp.c.

```
{
    lsgp_wait_for_detector.new_value_ready =
        0;
    pthread_mutex_init( &(lsgp_wait_for_detector.mutex
        ), NULL);
    pthread_cond_init( &(lsgp_wait_for_detector.cond),
        NULL);
}
```

#### 7.4.4.83 void lsgp\_wait\_for\_detector\_wait( )

Pause the calling thread until the detector is ready Called by the MD2 thread.

Definition at line 981 of file lsgp.c.

```
{
    pthread_mutex_lock( &(lsgp_wait_for_detector.mutex
        )));
    while( lsgp_wait_for_detector.new_value_ready
        == 0)
        pthread_cond_wait( &(lsgp_wait_for_detector.cond)
            , &(lsgp_wait_for_detector.mutex));
}
```

#### 7.4.4.84 void lsgp\_waitcryo\_all( )

no need to get fancy with the wait cryo command It should not return until the robot is almost ready for air rights

Definition at line 597 of file lsgp.c.

```
{
    pthread_mutex_lock( &lsgp_waitcryo.mutex);
    lsgp_waitcryo.new_value_ready = 0;

    lsgp_query_push( lsgp_waitcryo_cb, "SELECT
        px.waitcryo()");

    while( lsgp_waitcryo.new_value_ready == 0)
        pthread_cond_wait( &lsgp_waitcryo.cond, &lsgp_waitcryo
            .mutex);

    pthread_mutex_unlock( &lsgp_waitcryo.mutex);
}
```

7.4.4.85 void **lspg\_waitcryo\_cb** ( **lspg\_query\_queue\_t** \* *qqp*, **PGresult** \* *pgr* )

Definition at line 587 of file **lspg.c**.

```
{
pthread_mutex_lock( &lspg_waitcryo.mutex);
lspg_waitcryo.new_value_ready = 1;
pthread_cond_signal( &lspg_waitcryo.cond);
pthread_mutex_unlock( &lspg_waitcryo.mutex);
}
```

7.4.4.86 void **lspg\_waitcryo\_init** ( )

Definition at line 581 of file **lspg.c**.

```
{
lspg_waitcryo.new_value_ready = 0;
pthread_mutex_init( &lspg_waitcryo.mutex, NULL);
pthread_cond_init( &lspg_waitcryo.cond, NULL);
}
```

7.4.4.87 void\* **lspg\_worker** ( void \* *dummy* )

The main loop for the **lspg** thread.

## Parameters

in	<i>dummy</i>	Required by pthreads but unused
----	--------------	---------------------------------

Definition at line 1794 of file **lspg.c**.

```
{
static struct pollfd fda[2]; // 0=signal handler, 1=pg socket
static int nfda = 0;
static sigset_t our_sigset;

// 
// block ordinary signal mechanism
//
sigemptyset( &our_sigset);
sigaddset( &our_sigset, SIGUSR1);
pthread_sigmask(SIG_BLOCK, &our_sigset, NULL);

fda[0].fd = signalfd( -1, &our_sigset, SFD_NONBLOCK);
if( fda[0].fd == -1) {
    char *es;
    es = strerror( errno);
    lslogging_log_message( "Signalfd trouble: %s", es);
}
fda[0].events = POLLIN;

// 
// make sure file descriptor is not legal until it's been connected
//
lspgfd.fd = -1;

while( 1) {
    int pollrtn;
    int poll_timeout_ms;

    lspg_next_state();

    if( lspgfd.fd == -1) {
        //
        // Here a connection to the database is not established.
        // Periodically try again. Should possibly arrange to reconnect
        // to signalfd but that's unlikely to be necessary.
        //
        nfda = 1;
    }
}
```

```

    poll_timeout_ms = 10000;
    fda[1].revents = 0;
} else {
    //
    // Arrange to peacefully do nothing until either the pg server sends us
    // something
    // or someone pushes something onto our queue
    //
    nfda = 2;
    fda[1].fd      = lspgfd.fd;
    fda[1].events  = lspgfd.events;
    fda[1].revents = 0;
    poll_timeout_ms = -1;
}

pollrtn = poll( fda, nfda, poll_timeout_ms);

if( pollrtn && fda[0].revents) {
    lspg_sig_service( &(fda[0]));
    pollrtn--;
}
if( pollrtn && fda[1].revents) {
    lspg_pg_service( &(fda[1]));
    pollrtn--;
}
}
}

```

#### 7.4.5 Variable Documentation

7.4.5.1 int ls\_pg\_state = LS\_PG\_STATE\_INIT [static]

## State of the Ispg state machine.

Definition at line 38 of file lspg.c.

**7.4.5.2 PostgresPollingStatusType lspgsql\_connectPoll\_response [static]**

Used to determine state while connecting.

Definition at line 59 of file lspg.c.

#### 7.4.5.3 `lspg_demandairrights` t `lspg_demandairrights`

our demandairrights object

Definition at line 65 of file lspg.c.

#### 7.4.5.4 lspg\_getcenter

the getcenter object

Definition at line 64 of file lspq.c.

#### 7.4.5.5 `lspg_getcurrentsampleid_t` `lspg_getcurrentsampleid`

our current sample id

Definition at line 66 of file lspq.c.

#### 7.4.5.6 **lspg\_lock\_detector\_t** **lspg\_lock\_detector** [static]

Definition at line 1070 of file lsgc.c.

**7.4.5.7 lspg\_lock\_diffractometer\_t lspg\_lock\_diffractometer [static]**

Definition at line 1011 of file lspg.c.

**7.4.5.8 lspg\_nexsample\_t lspg\_nexsample**

the very next sample

Definition at line 62 of file lspg.c.

**7.4.5.9 lspg\_nextshot\_t lspg\_nextshot**

the nextshot object

Definition at line 63 of file lspg.c.

**7.4.5.10 lspg\_query\_queue\_t lspg\_query\_queue[LS\_PG\_QUERY\_QUEUE\_LENGTH] [static]**

Our query queue.

Definition at line 51 of file lspg.c.

**7.4.5.11 unsigned int lspg\_query\_queue\_off = 0 [static]**

The last item still being used (on == off means nothing in queue)

Definition at line 53 of file lspg.c.

**7.4.5.12 unsigned int lspg\_query\_queue\_on = 0 [static]**

Next position to add something to the queue.

Definition at line 52 of file lspg.c.

**7.4.5.13 unsigned int lspg\_query\_queue\_reply = 0 [static]**

The current item being digested.

Normally off <= reply <= on. Corner case of queue wrap around works because we only increment and compare for equality.

Definition at line 54 of file lspg.c.

**7.4.5.14 pthread\_cond\_t lspg\_query\_queue\_cond [static]**

keeps the queue from overflowing

Definition at line 43 of file lspg.c.

**7.4.5.15 pthread\_mutex\_t lspg\_query\_queue\_mutex [static]**

keep the queue from getting tangled

Definition at line 42 of file lspg.c.

**7.4.5.16 PostgresPollingStatusType lsgp\_resetPoll\_response [static]**

Used to determine state while reconnecting.

Definition at line 60 of file lsgp.c.

**7.4.5.17 lsgp\_seq\_run\_prep\_t lsgp\_seq\_run\_prep [static]**

Definition at line 1128 of file lsgp.c.

**7.4.5.18 lsgp\_starttransfer\_t lsgp\_starttransfer**

start a sample transfer

Definition at line 67 of file lsgp.c.

**7.4.5.19 pthread\_t lsgp\_thread [static]**

our worker thread

Definition at line 41 of file lsgp.c.

**7.4.5.20 lsgp\_wait\_for\_detector\_t lsgp\_wait\_for\_detector [static]**

Instance of the detector timing object.

Definition at line 949 of file lsgp.c.

**7.4.5.21 lsgp\_waitcryo\_t lsgp\_waitcryo**

signal the robot

Definition at line 68 of file lsgp.c.

**7.4.5.22 struct pollfd lsgpfdf [static]**

our poll info

Definition at line 44 of file lsgp.c.

**7.4.5.23 struct timeval lsgp\_time\_sent now [static]**

used to ensure we do not inundate the db server with connection requests

Definition at line 39 of file lsgp.c.

**7.4.5.24 PGconn\* q = NULL [static]**

Database connector.

Definition at line 58 of file lsgp.c.

## 7.5 lspmac.c File Reference

Routines concerned with communication with PMAC.

```
#include "pgpmac.h"
```

### Data Structures

- struct `md2StatusStruct`  
*The block of memory retrieved in a status request.*
- struct `lspmac_ascii_buffers_struct`
- struct `lspmac_dpascii_queue_struct`
- struct `lspmac_combined_move_struct`

### Macros

- `#define LS_PMAC_STATE_RESET -1`
- `#define LS_PMAC_STATE_DETACHED 0`
- `#define LS_PMAC_STATE_IDLE 1`
- `#define LS_PMAC_STATE_SC 2`
- `#define LS_PMAC_STATE_WACK_NFR 3`
- `#define LS_PMAC_STATE_WACK_CC 4`
- `#define LS_PMAC_STATE_WACK 5`
- `#define LS_PMAC_STATE_GMR 6`
- `#define LS_PMAC_STATE_CR 7`
- `#define LS_PMAC_STATE_RR 8`
- `#define LS_PMAC_STATE_WACK_RR 9`
- `#define LS_PMAC_STATE_GB 10`
- `#define LS_PMAC_STATE_WCR 11`
- `#define LS_PMAC_STATE_WGB 12`
- `#define LSPMAC_MAX_MOTORS 48`
- `#define LSPMAC_PRESET_REGEX "(.*\\.%s\\.\presets)\\.(0-9]+)\\.(name|position)"`  
*Regex to pick out preset name and corresponding position.*

- `#define PMACPORT 1025`  
*The PMAC (only) listens on this port.*
- `#define pmac_cmd_size 8`  
*PMAC command size in bytes.*
- `#define VR_UPLOAD 0xc0`
- `#define VR_DOWNLOAD 0x40`
- `#define VR_PMAC_SENDBUF 0xb0`
- `#define VR_PMAC_GETLINE 0xb1`
- `#define VR_PMAC_FLUSH 0xb3`
- `#define VR_PMAC_GETMEM 0xb4`
- `#define VR_PMAC_SETMEM 0xb5`
- `#define VR_PMAC_SENDCTRLCHAR 0xb6`
- `#define VR_PMAC_SETBIT 0xba`
- `#define VR_PMAC_SETBITS 0xbb`
- `#define VR_PMAC_PORT 0xbe`
- `#define VR_PMAC_GETRESPONSE 0xbf`
- `#define VR_PMAC_READREADY 0xc2`
- `#define VR_CTRL_RESPONSE 0xc4`
- `#define VR_PMAC_GETBUFFER 0xc5`

- #define VR\_PMAC\_WRITEBUFFER 0xc6
- #define VR\_PMAC\_WRITEERROR 0xc7
- #define VR\_FWDOWNLOAD 0xcb
- #define VR\_IPADDRESS 0xe0
- #define PMAC\_MIN\_CMD\_TIME 10000.0
 

*Minimum time between commands to the pmac.*
- #define PMAC\_CMD\_QUEUE\_LENGTH 2048
 

*Size of the PMAC command queue.*
- #define LSPMAC\_DPASCII\_QUEUE\_LENGTH 1024

## Typedefs

- typedef struct md2StatusStruct md2\_status\_t
 

*The block of memory retrieved in a status request.*
- typedef struct lspmac\_ascii\_buffers\_struct lspmac\_ascii\_buffers\_t
- typedef struct lspmac\_dpascii\_queue\_struct lspmac\_dpascii\_queue\_t
- typedef struct lspmac\_combined\_move\_struct lspmac\_combined\_move\_t

## Functions

- void **lspmac\_get\_ascii** (char \*)
 

*Forward declareation.*
- double **lspmac\_lut** (int nlut, double \*lut, double x)
 

*Look up table support for motor positions (think x=zoom, y=light intensity) use a lookup table to find the "counts" to move the motor to the requested position The look up table is a simple one dimensional array with the x values as even indicies and the y values as odd indices.*
- double **lspmac\_rlut** (int nlut, double \*lut, double y)
- void **hex\_dump** (int n, unsigned char \*s)
 

*Prints a hex dump of the given data.*
- char \* **cleanstr** (char \*s)
 

*Replace \r with \n in null terminated string and print result to terminal.*
- void **IsConnect** (char \*ipaddr)
 

*Connect to the PMAC socket.*
- void **lspmac\_reset\_queue** ()
 

*Clear the queue as part of PMAC reinitialization.*
- **pmac\_cmd\_queue\_t \* lspmac\_push\_queue (pmac\_cmd\_queue\_t \*cmd)**

*Put a new command on the queue.*
- **pmac\_cmd\_queue\_t \* lspmac\_pop\_queue ()**

*Remove the oldest queue item.*
- **pmac\_cmd\_queue\_t \* lspmac\_pop\_reply ()**

*Remove the next command queue item that is waiting for a reply.*
- **pmac\_cmd\_queue\_t \* lspmac\_send\_command (int rqType, int rq, int wValue, int wlIndex, int wLength, char \*data, void(\*responseCB)(pmac\_cmd\_queue\_t \*, int, char \*), int no\_reply, char \*event)**

*Compose a packet and send it to the PMAC.*
- void **lspmac\_SockFlush** ()
 

*Reset the PMAC socket from the PMAC side.*
- void **lspmac\_Reset** ()
 

*Clear the queue and put the PMAC into a known state.*
- void **lspmac\_Error** (char \*buff)

- The service routing detected an error condition.*
- void `lspmac_Service` (struct pollfd \*evt)
 

*Service routine for packet coming from the PMAC.*
  - void `lspmac_GetShortReplyCB` (pmac\_cmd\_queue\_t \*cmd, int nreceived, char \*buff)
 

*Receive a reply that does not require multiple buffers.*
  - void `lspmac_SendControlReplyPrintCB` (pmac\_cmd\_queue\_t \*cmd, int nreceived, char \*buff)
 

*Receive a reply to a control character Print a "printable" version of the character to the terminal Followed by a hex dump of the response.*
  - void `lspmac_GetmemReplyCB` (pmac\_cmd\_queue\_t \*cmd, int nreceived, char \*buff)
 

*Service a reply to the getmem command.*
  - pmac\_cmd\_queue\_t \* `lspmac_SockGetmem` (int offset, int nbytes)
 

*Request a chunk of memory to be returned.*
  - pmac\_cmd\_queue\_t \* `lspmac_SockSendline` (char \*event, char \*fmt,...)
 

*Send a one line command.*
  - pmac\_cmd\_queue\_t \* `lspmac_SockSendline_nr` (char \*event, char \*fmt,...)
 

*Send a command and ignore the response.*
  - pmac\_cmd\_queue\_t \* `lspmac_SockSendControlCharPrint` (char \*event, char c)
 

*Send a control character.*
  - void `lspmac_Getmem` ()
 

*Request a block of double buffer memory.*
  - void `lspmac_bo_read` (lspmac\_motor\_t \*mp)
 

*Read the state of a binary i/o motor This is the read method for the binary i/o motor class.*
  - void `lspmac_dac_read` (lspmac\_motor\_t \*mp)
 

*Read a DAC motor position.*
  - void `lspmac_shutter_read` (lspmac\_motor\_t \*mp)
 

*Fast shutter read routine The shutter is mildly complicated in that we need to take into account the fact that the shutter can open and close again between status updates.*
  - void `lspmac_home1_queue` (lspmac\_motor\_t \*mp)
 

*Home the motor.*
  - void `lspmac_home2_queue` (lspmac\_motor\_t \*mp)
 

*Second stage of homing.*
  - double `lspmac_getPosition` (lspmac\_motor\_t \*mp)
 

*get the motor position (with locking)*
  - void `lspmac_pmacmotor_read` (lspmac\_motor\_t \*mp)
 

*Read the position and status of a normal PMAC motor.*
  - int `lspmac_getBIPosition` (lspmac\_bi\_t \*bip)
 

*get binary input value*
  - void `lspmac_get_status_cb` (pmac\_cmd\_queue\_t \*cmd, int nreceived, char \*buff)
 

*Service routing for status update This updates positions and status information.*
  - void `lspmac_get_status` ()
 

*Request a status update from the PMAC.*
  - void `lspmac_more_ascii_cb` (pmac\_cmd\_queue\_t \*cmd, int nreceived, char \*buff)
 

*we are expecting more characters from the DPRAM ASCII interface*
  - void `lspmac_get_ascii_cb` (pmac\_cmd\_queue\_t \*cmd, int nreceived, char \*buff)
 

*service the ascii buffer request response*
  - void `lspmac_asciicmdCB` (pmac\_cmd\_queue\_t \*cmd, int nreceived, char \*buf)
 

*PMAC has received our ascii command request Now see when it is ready for the next one.*
  - void `lspmac_SockSendDPLine` (char \*event, char \*fmt,...)
 

*prepare (queue up) a line to send the dpram ascii command interface*
  - void `lspmac_request_control_response_cb` (char \*event)
  - void `lspmac_SockSendDPControlCharCB` (pmac\_cmd\_queue\_t \*cmd, int nreceived, char \*buf)

- void `Ispmac_SockSendDPControlChar` (char \*event, char c)
 

*use dpram ascii interface to send a control character*
- void `Ispmac_SockSendDPqueue` ()
- void `Ispmac_abort` ()
 

*abort motion and try to recover*
- void `Ispmac_GetAllIVarsCB` (pmac\_cmd\_queue\_t \*cmd, int nreceived, char \*buff)
 

*Receive the values of all the I variables Update our Postgresql database with the results.*
- void `Ispmac_GetAllIVars` ()
 

*Request the values of all the I variables.*
- void `Ispmac_GetAllMVarsCB` (pmac\_cmd\_queue\_t \*cmd, int nreceived, char \*buff)
 

*Receive the values of all the M variables Update our database with the results.*
- void `Ispmac_GetAllMVars` ()
 

*Request the values of all the M variables.*
- void `Ispmac_sendcmd_nocb` (char \*fmt,...)
 

*Send a command that does not need to deal with the reply.*
- void `Ispmac_sendcmd` (char \*event, void(\*responseCB)(pmac\_cmd\_queue\_t \*, int, char \*), char \*fmt,...)
 

*PMAC command with call back.*
- void `Ispmac_next_state` ()
 

*State machine logic.*
- void \* `Ispmac_worker` (void \*dummy)
 

*Our Ispmac worker thread.*
- int `Ispmac_movedac_queue` (Ispmac\_motor\_t \*mp, double requested\_position)
 

*Move method for dac motor objects (ie, lights)*
- int `Ispmac_movezoom_queue` (Ispmac\_motor\_t \*mp, double requested\_position)
 

*Move method for the zoom motor.*
- int `Ispmac_move_preset_queue` (Ispmac\_motor\_t \*mp, char \*preset\_name)
 

*Move a given motor to one of its preset positions.*
- int `Ispmac_test_preset` (Ispmac\_motor\_t \*mp, char \*preset\_name, double tolerance)
 

*see if the motor is within tolerance of the preset 1 means yes, it is 0 mean no it isn't or that the preset was not found*
- int `Ispmac_moveabs_fshut_queue` (Ispmac\_motor\_t \*mp, double requested\_position)
 

*Move method for the fast shutter.*
- int `Ispmac_moveabs_bo_queue` (Ispmac\_motor\_t \*mp, double requested\_position)
 

*Move method for binary i/o motor objects.*
- void `Ispmac_moveabs_timed_queue` (Ispmac\_motor\_t \*mp, double start, double delta, double time)
 

*timed motor move*
- int `Ispmac_moveabs_frontlight_oo_queue` (Ispmac\_motor\_t \*mp, double pos)
 

*"move" frontlight on/off*
- int `Ispmac_moveabs_flight_factor_queue` (Ispmac\_motor\_t \*mp, double pos)
- int `Ispmac_moveabs_blight_factor_queue` (Ispmac\_motor\_t \*mp, double pos)
- void `Ispmac_video_rotate` (double secs)
 

*Special motion program to collect centering video.*
- int `Ispmac_set_motion_flags` (int \*mmaskp, Ispmac\_motor\_t \*mp\_1,...)
 

*Set the coordinate system motion flags (m5075) for the null terminated list of motors that we are planning on running a motion program with.*
- int `Ispmac_est_move_time` (double \*est\_time, int \*mmaskp, Ispmac\_motor\_t \*mp\_1, int jog\_1, char \*preset\_1, double end\_point\_1,...)
 

*Move the motors and estimate the time it'll take to finish the job.*
- int `Ispmac_est_move_time_wait` (double move\_time, int cmask, Ispmac\_motor\_t \*mp\_1,...)
 

*wait for motion to stop returns non-zero if the wait timed out*
- int `Ispmac_move_or_jog_abs_queue` (Ispmac\_motor\_t \*mp, double requested\_position, int use\_jog)
 

*Move method for normal stepper and servo motor objects Returns non-zero on abort, zero if OK.*

- int `lsmpmac_move_or_jog_preset_queue (lsmpmac_motor_t *mp, char *preset, int use_jog)`  
*move using a preset value returns 0 on success, non-zero on error*
- int `lsmpmac_moveabs_queue (lsmpmac_motor_t *mp, double requested_position)`  
*Use coordinate system motion program, if available, to move motor to requested position.*
- int `lsmpmac_jogabs_queue (lsmpmac_motor_t *mp, double requested_position)`  
*Use jog to move motor to requested position.*
- int `lsmpmac_moveabs_wait (lsmpmac_motor_t *mp, double timeout_secs)`  
*Wait for motor to finish moving.*
- void `_lsmpmac_motor_init (lsmpmac_motor_t *d, char *name)`  
*Helper funciton for the init calls.*
- `lsmpmac_motor_t * lsmpmac_motor_init (lsmpmac_motor_t *d, int wy, int wx, int *posp, int *stat1p, int *stat2p, char *wtitle, char *name, int(*moveAbs)(lsmpmac_motor_t *, double), int(*jogAbs)(lsmpmac_motor_t *, double))`  
*Initialize a pmac stepper or servo motor.*
- `lsmpmac_motor_t * lsmpmac_fshut_init (lsmpmac_motor_t *d)`  
*Initialize the fast shutter motor.*
- `lsmpmac_motor_t * lsmpmac_bo_init (lsmpmac_motor_t *d, char *name, char *write_fmt, int *read_ptr, int read_mask)`  
*Initialize binary i/o motor.*
- `lsmpmac_motor_t * lsmpmac_dac_init (lsmpmac_motor_t *d, int *posp, char *mvar, char *name, int(*moveAbs)(lsmpmac_motor_t *, double))`  
*Initialize DAC motor Note that some motors require further initialization from a database query.*
- void `lsmpmac_soft_motor_read (lsmpmac_motor_t *p)`  
*Dummy routine to read a soft motor.*
- `lsmpmac_motor_t * lsmpmac_soft_motor_init (lsmpmac_motor_t *d, char *name, int(*moveAbs)(lsmpmac_motor_t *, double))`
- `lsmpmac_bi_t * lsmpmac_bi_init (lsmpmac_bi_t *d, int *ptr, int mask, char *onEvent, char *offEvent)`  
*Initialize binary input.*
- void `lsmpmac_full_card_reset_cb (char *event)`  
*reset and restart*
- void `lsmpmac_init (int ivarsflag, int mvarsflag)`  
*Initialize this module.*
- void `lsmpmac_cryoSwitchChanged_cb (char *event)`
- void `lsmpmac_scint_maybe_turn_on_dryer_cb (char *event)`  
*Maybe start drying off the scintilator.*
- void `lsmpmac_scint_maybe_turn_off_dryer_cb (char *event)`  
*Maybe stop drying off the scintilator.*
- void `lsmpmac_backLight_up_cb (char *event)`  
*Turn on the backlight whenever it goes up.*
- void `lsmpmac_backLight_down_cb (char *event)`  
*Turn off the backlight whenever it goes down.*
- void `lsmpmac_light_zoom_cb (char *event)`  
*Set the backlight intensity whenever the zoom is changed (and the backlight is up)*
- void `lsmpmac_quitting_cb (char *event)`  
*prepare to exit program in a couple of seconds*
- void `lsmpmac_scint_maybe_move_sample_cb (char *event)`  
*Perhaps we need to move the sample out of the way.*
- void `lsmpmac_scint_maybe_return_sample_cb (char *event)`  
*Perhaps we need to return the sample to the beam.*
- void `lsmpmac_scint_dried_cb (char *event)`  
*Turn off the dryer.*

- void `lspmac_zoom_lut_setup ()`  
*Set up lookup table for zoom.*
- void `lspmac_flight_lut_setup ()`  
*Set up lookup table for flight.*
- void `lspmac_blight_lut_setup ()`  
*Set up lookup table for blight.*
- void `lspmac_fscint_lut_setup ()`  
*Set up lookup table for fscint.*
- `lspmac_motor_t * lspmac_find_motor_by_name (char *name)`
- void `lspmac_command_done_cb (char *event)`
- void `lspmac_run ()`  
*Start up the lspmac thread.*

## Variables

- static int `ls_pmac_state = LS_PMAC_STATE_DETACHED`  
*Current state of the PMAC communications state machine.*
- static int `lspmac_running = 1`  
*exit worker thread when zero*
- int `lspmac_shutter_state`  
*State of the shutter, used to detect changes.*
- int `lspmac_shutter_has_opened`  
*Indicates that the shutter had opened, perhaps briefly even if the state did not change.*
- pthread\_mutex\_t `lspmac_shutter_mutex`  
*Coordinates threads reading shutter status.*
- pthread\_cond\_t `lspmac_shutter_cond`  
*Allows waiting for the shutter status to change.*
- pthread\_mutex\_t `lspmac_moving_mutex`  
*Coordinate moving motors between threads.*
- pthread\_cond\_t `lspmac_moving_cond`  
*Wait for motor(s) to finish moving condition.*
- int `lspmac_moving_flags`  
*Flag used to implement motor moving condition.*
- static uint16\_t `lspmac_control_char = 0`  
*The control character we've sent.*
- static pthread\_mutex\_t `lspmac_ascii_mutex`  
*Keep too many processes from sending commands at once.*
- static int `lspmac_ascii_busy = 0`  
*flag for condition to wait for*
- static int `omega_zero_search = 0`  
*Indicate we'd really like to know when omega crosses zero.*
- static double `omega_zero_velocity = 0`  
*rate (cnts/sec) that omega was traveling when it crossed zero*
- struct timespec `omega_zero_time`  
*Time we believe that omega crossed zero.*
- static struct timespec `lspmac_status_time`  
*Time the status was read.*
- static struct timespec `lspmac_status_last_time`  
*Time the status was read.*
- static pthread\_t `pmac_thread`

- `pthread_mutex_t pmac_queue_mutex`  
*our thread to manage access and communication to the pmac*
- `pthread_cond_t pmac_queue_cond`  
*manage access to the pmac command queue*
- `pthread_cond_t pmac_queue_cond`  
*wait for a command to be sent to PMAC before continuing*
- `static struct pollfd pmacfd`  
*our poll structure*
- `static int getivars = 0`  
*flag set at initialization to send i vars to db*
- `static int getmvars = 0`  
*flag set at initialization to send m vars to db*
- `lspmac_bi_t lspmac_bis [32]`  
*array of binary inputs*
- `int lspmac_nbis = 0`  
*number of active binary inputs*
- `lspmac_motor_t lspmac_motors [LSPMAC_MAX_MOTORS]`  
*All our motors.*
- `int lspmac_nmotors = 0`  
*The number of motors we manage.*
- `struct hsearch_data motors_ht`  
*A hash table to find motors by name.*
- `lspmac_motor_t * omega`  
*MD2 omega axis (the air bearing)*
- `lspmac_motor_t * alignx`  
*Alignment stage X.*
- `lspmac_motor_t * aligny`  
*Alignment stage Y.*
- `lspmac_motor_t * alignz`  
*Alignment stage Z.*
- `lspmac_motor_t * anal`  
*Polaroid analyzer motor.*
- `lspmac_motor_t * zoom`  
*Optical zoom.*
- `lspmac_motor_t * apery`  
*Aperture Y.*
- `lspmac_motor_t * aperz`  
*Aperture Z.*
- `lspmac_motor_t * capy`  
*Capillary Y.*
- `lspmac_motor_t * capz`  
*Capillary Z.*
- `lspmac_motor_t * scint`  
*Scintillator Z.*
- `lspmac_motor_t * cenx`  
*Centering Table X.*
- `lspmac_motor_t * ceny`  
*Centering Table Y.*
- `lspmac_motor_t * kappa`  
*Kappa.*
- `lspmac_motor_t * phi`  
*Phi (not data collection axis)*

- `lspmac_motor_t * fshut`  
*Fast shutter.*
- `lspmac_motor_t * flight`  
*Front Light DAC.*
- `lspmac_motor_t * blight`  
*Back Light DAC.*
- `lspmac_motor_t * fscint`  
*Scintillator Piezo DAC.*
- `lspmac_motor_t * smart_mag_oo`  
*Smart Magnet on/off.*
- `lspmac_motor_t * blight_ud`  
*Back light Up/Down actuator.*
- `lspmac_motor_t * cryo`  
*Move the cryostream towards or away from the crystal.*
- `lspmac_motor_t * dryer`  
*blow air on the scintilator to dry it off*
- `lspmac_motor_t * fluo`  
*Move the fluorescence detector in/out.*
- `lspmac_motor_t * flight_oo`  
*Turn front light on/off.*
- `lspmac_motor_t * blight_f`  
*Back light scale factor.*
- `lspmac_motor_t * flight_f`  
*Front light scale factor.*
- `lspmac_bi_t * lp_air`  
*Low pressure air OK.*
- `lspmac_bi_t * hp_air`  
*High pressure air OK.*
- `lspmac_bi_t * cryo_switch`  
*that little toggle switch for the cryo*
- `lspmac_bi_t * blight_down`  
*Backlight is down.*
- `lspmac_bi_t * blight_up`  
*Backlight is up.*
- `lspmac_bi_t * cryo_back`  
*cryo is in the back position*
- `lspmac_bi_t * fluor_back`  
*fluor is in the back position*
- `lspmac_bi_t * sample_detected`  
*smart magnet detected sample*
- `lspmac_bi_t * etel_ready`  
*ETEL is ready.*
- `lspmac_bi_t * etel_on`  
*ETEL is on.*
- `lspmac_bi_t * etel_init_ok`  
*ETEL initialized OK.*
- `lspmac_bi_t * minikappa_ok`  
*Minikappa is OK (whatever that means)*
- `lspmac_bi_t * smart_mag_on`  
*smart magnet is on*
- `lspmac_bi_t * arm_parked`

- (whose arm? parked where?)
- **lspmac.bi\_t \* shutter\_open**  
shutter is open (note in pmc says this is a slow input)
- **lspmac.bi\_t \* smart\_mag\_err**  
smart magnet error (coil broken perhaps)
- **lspmac.bi\_t \* smart\_mag\_off**  
smart magnet is off
- static unsigned char **dbmem** [64 \*1024]  
double buffered memory
- static int **dbmemIn** = 0  
next location
- static struct timeval  
pmac\_time\_sent **now**  
used to ensure we do not send commands to the pmac too often. Only needed for non-DB commands.
- static **pmac\_cmd\_t rr\_cmd**
- static **pmac\_cmd\_t gb\_cmd**
- static **pmac\_cmd\_t cr\_cmd**  
commands to send out "readready", "getbuffer", "controlresponse" (initialized in main)
- static **pmac\_cmd\_queue\_t ethCmdQueue** [**PMAC\_CMD\_QUEUE\_LENGTH**]  
PMAC command queue.
- static unsigned int **ethCmdOn** = 0  
points to next empty PMAC command queue position
- static unsigned int **ethCmdOff** = 0  
points to current command (or none if == ethCmdOn)
- static unsigned int **ethCmdReply** = 0  
Used like ethCmdOff only to deal with the pmac reply to a command.
- static char \* **pmac\_error\_strs** []  
Decode the errors perhaps returned by the PMAC.
- static **md2\_status\_t md2\_status**  
Buffer for MD2 Status.
- **pthread\_mutex\_t md2\_status\_mutex**  
Synchronize reading/writing status buffer.
- static **lspmac\_ascii\_buffers\_t lspmac\_ascii\_buffers**
- **pthread\_mutex\_t lspmac\_ascii\_buffers\_mutex**
- static **lspmac\_dpascii\_queue\_t lspmac\_dpascii\_queue** [**LSPMAC\_DPASCII\_QUEUE\_LENGTH**]
- static uint32\_t **lspmac\_dpascii\_on** = 0
- static uint32\_t **lspmac\_dpascii\_off** = 0

### 7.5.1 Detailed Description

Routines concerned with communication with PMAC. Test suite for the pgpmac routines.

```
\date 2012 - 2013
\author Keith Brister
\copyright All Rights Reserved
```

This is a state machine (surprise!)

Lacking support for writingbuffer, control writing and reading, as well as double buffered memory It looks like several different methods of managing PMAC communications are possible. Here is set up a queue of outgoing commands and deal completely with the result before sending the next. A full handshake of acknowledgements and "readready" is expected.

Most of these states are to deal with the "serial-port" style of communications. Things are surprisingly simple for the double buffer ascii and control character methods.

State	Description
-1	Reset the connection
0	Detached: need to connect to tcp port
1	Idle (waiting for a command to send to the pmac)
2	Send command
3	Waiting for command acknowledgement (no further response expected)
4	Waiting for control character acknowledgement (further response expected)
5	Waiting for command acknowledgement (further response expected)
6	Waiting for get memory response
7	Send controlresponse
8	Send readready
9	Waiting for acknowledgement of "readready"
10	Send readbuffer
11	Waiting for control response
12	Waiting for readbuffer response

**Date**

2013

**Author**

Keith Brister

**Copyright**

All Rights Reserved

A place to put unit tests.

Definition in file [lspmac.c](#).

## 7.5.2 Macro Definition Documentation

### 7.5.2.1 #define LS\_PMAC\_STATE\_CR 7

Definition at line 52 of file [lspmac.c](#).

### 7.5.2.2 #define LS\_PMAC\_STATE\_DETACHED 0

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

### 7.5.2.3 #define LS\_PMAC\_STATE\_GB 10

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

### 7.5.2.4 #define LS\_PMAC\_STATE\_GMR 6

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

7.5.2.5 #define LS\_PMAC\_STATE\_IDLE 1

Definition at line 46 of file lspmac.c.

7.5.2.6 #define LS\_PMAC\_STATE\_RESET -1

Definition at line 44 of file lspmac.c.

7.5.2.7 #define LS\_PMAC\_STATE\_RR 8

Definition at line 53 of file lspmac.c.

7.5.2.8 #define LS\_PMAC\_STATE\_SC 2

Definition at line 47 of file lspmac.c.

7.5.2.9 #define LS\_PMAC\_STATE\_WACK 5

Definition at line 50 of file lspmac.c.

7.5.2.10 #define LS\_PMAC\_STATE\_WACK\_CC 4

Definition at line 49 of file lspmac.c.

7.5.2.11 #define LS\_PMAC\_STATE\_WACK\_NFR 3

Definition at line 48 of file lspmac.c.

7.5.2.12 #define LS\_PMAC\_STATE\_WACK\_RR 9

Definition at line 54 of file lspmac.c.

7.5.2.13 #define LS\_PMAC\_STATE\_WCR 11

Definition at line 56 of file lspmac.c.

7.5.2.14 #define LS\_PMAC\_STATE\_WGB 12

Definition at line 57 of file lspmac.c.

7.5.2.15 #define LSPMAC\_DPASCII\_QUEUE\_LENGTH 1024

Definition at line 370 of file lspmac.c.

7.5.2.16 #define LSPMAC\_MAX\_MOTORS 48

Definition at line 97 of file lspmac.c.

7.5.2.17 #define LSPMAC\_PRESET\_REGEX ".\*\\.%s\\\\.presets)\\\\.([0-9]+)\\\\.\\.(name|position)"

Regex to pick out preset name and corresponding position.

Definition at line 153 of file lspmacc.c.

7.5.2.18 #define PMAC\_CMD\_QUEUE\_LENGTH 2048

Size of the PMAC command queue.

Definition at line 197 of file lspmacc.c.

7.5.2.19 #define pmac\_cmd\_size 8

PMAC command size in bytes.

Definition at line 163 of file lspmacc.c.

7.5.2.20 #define PMAC\_MIN\_CMD\_TIME 10000.0

Minimum time between commands to the pmac.

Definition at line 193 of file lspmacc.c.

7.5.2.21 #define PMACPORT 1025

The PMAC (only) listens on this port.

Definition at line 157 of file lspmacc.c.

7.5.2.22 #define VR\_CTRL\_RESPONSE 0xc4

Definition at line 179 of file lspmacc.c.

7.5.2.23 #define VR\_DOWNLOAD 0x40

Definition at line 166 of file lspmacc.c.

7.5.2.24 #define VR\_FWDOWNLOAD 0xcb

Definition at line 183 of file lspmacc.c.

7.5.2.25 #define VR\_IPADDRESS 0xe0

Definition at line 184 of file lspmacc.c.

7.5.2.26 #define VR\_PMAC\_FLUSH 0xb3

Definition at line 170 of file lspmacc.c.

7.5.2.27 #define VR\_PMAC\_GETBUFFER 0xc5

Definition at line 180 of file lspmacc.c.

7.5.2.28 #define VR\_PMAC\_GETLINE 0xb1

Definition at line 169 of file lspmac.c.

7.5.2.29 #define VR\_PMAC\_GETMEM 0xb4

Definition at line 171 of file lspmac.c.

7.5.2.30 #define VR\_PMAC\_GETRESPONSE 0xbf

Definition at line 177 of file lspmac.c.

7.5.2.31 #define VR\_PMAC\_PORT 0xbe

Definition at line 176 of file lspmac.c.

7.5.2.32 #define VR\_PMAC\_READREADY 0xc2

Definition at line 178 of file lspmac.c.

7.5.2.33 #define VR\_PMAC\_SENDCTRLCHAR 0xb6

Definition at line 173 of file lspmac.c.

7.5.2.34 #define VR\_PMAC\_SENDLINE 0xb0

Definition at line 168 of file lspmac.c.

7.5.2.35 #define VR\_PMAC\_SETBIT 0xba

Definition at line 174 of file lspmac.c.

7.5.2.36 #define VR\_PMAC\_SETBITS 0xbb

Definition at line 175 of file lspmac.c.

7.5.2.37 #define VR\_PMAC\_SETMEM 0xb5

Definition at line 172 of file lspmac.c.

7.5.2.38 #define VR\_PMAC\_WRITEBUFFER 0xc6

Definition at line 181 of file lspmac.c.

7.5.2.39 #define VR\_PMAC\_WRITEERROR 0xc7

Definition at line 182 of file lspmac.c.

### 7.5.2.40 #define VR\_UPLOAD 0xc0

Definition at line 165 of file lspmac.c.

## 7.5.3 Typedef Documentation

### 7.5.3.1 typedef struct lspmac\_ascii\_buffers\_struct lspmac\_ascii\_buffers\_t

### 7.5.3.2 typedef struct lspmac\_combined\_move\_struct lspmac\_combined\_move\_t

### 7.5.3.3 typedef struct lspmac\_dpascii\_queue\_struct lspmac\_dpascii\_queue\_t

### 7.5.3.4 typedef struct md2StatusStruct md2\_status\_t

The block of memory retrieved in a status request.

## 7.5.4 Function Documentation

### 7.5.4.1 void lspmac\_motor\_init( lspmac\_motor\_t \* d, char \* name )

Helper function for the init calls.

Definition at line 3573 of file lspmac.c.

```

{
pthread_mutexattr_t mutex_initializer;
// Use recursive mutexs
//
pthread_mutexattr_init( &mutex_initializer );
pthread_mutexattr_settype( &mutex_initializer, PTHREAD_MUTEX_RECURSIVE );

lspmac_nmotors++;

pthread_mutex_init( &(d->mutex), &mutex_initializer );
pthread_cond_init( &(d->cond), NULL );

d->magic          = LSPMAC_MAGIC_NUMBER;
d->name           = strdup(name);
d->active         = lsredis_get_obj( "%s.active",
                                       d->name );
d->active_init    = lsredis_get_obj( "%s.active_init",
                                       d->name );
d->axis            = lsredis_get_obj( "%s.axis",
                                       d->name );
d->coord_num      = lsredis_get_obj( "%s.coord_num",
                                       d->name );
d->home            = lsredis_get_obj( "%s.home",
                                       d->name );
d->in_position_band = lsredis_get_obj( "%s.in_position_band",
                                         d->name );
d->inactive_init   = lsredis_get_obj( "%s.inactive_init",
                                         d->name );
d->redis_fmt       = lsredis_get_obj( "%s.format",
                                         d->name );
d->max_accel       = lsredis_get_obj( "%s.max_accel",
                                         d->name );
d->max_speed        = lsredis_get_obj( "%s.max_speed",
                                         d->name );
d->max_pos          = lsredis_get_obj( "%s.maxPosition",
                                         d->name );
d->min_pos          = lsredis_get_obj( "%s.minPosition",
                                         d->name );
d->motor_num        = lsredis_get_obj( "%s.motor_num",
                                         d->name );
d->neg_limit_hit   = lsredis_get_obj( "%s.negLimitSet",
                                         d->name );
d->neutral_pos     = lsredis_get_obj( "%s.neutralPosition",
                                         d->name );
d->redis_position   = lsredis_get_obj( "%s.position",
                                         d->name );
d->pos_limit_hit   = lsredis_get_obj( "%s.posLimitSet",
                                         d->name );
d->precision        = lsredis_get_obj( "%s.precision",
                                         d->name );
}

```

```

    "%s.precision",      d->name);
d->printf_fmt        = lsredis_get_obj( " 
    "%s.printf",
d->status_str         = lsredis_get_obj( "
    "%s.status_str",
d->u2c                = lsredis_get_obj( "%s.u2c",
    d->name);
d->unit                = lsredis_get_obj( "%s.unit",
    d->name);
d->update_resolution   = lsredis_get_obj( "
    "%s.update_resolution", d->name);
d->lut                  = NULL;
d->nlut                 = 0;
d->homing                = 0;
d->dac_mvar               = NULL;
d->actual_pos_ctns_p     = NULL;
d->status1_p              = NULL;
d->status2_p              = NULL;
d->win                   = NULL;
d->read                  = NULL;
d->reported_position     = INFINITY;
d->reported_pg_position= INFINITY;

lsevents_preregister_event( "%s queued", d->name
);
lsevents_preregister_event( "%s command accepted",
d->name);

lsredis_load_presets( d->name);
}

```

#### 7.5.4.2 char\* cleanstr( char \* s )

Replace \r with \n in null terminated string and print result to terminal.

Needed to turn PMAC messages into something printable.

##### Parameters

in	s	String to print to terminal.
----	---	------------------------------

Definition at line 556 of file lspmac.c.

```

{
char t[256];
int i;

t[0] = 0;
for( i=0; i<strlen( s) && i < sizeof( t); i++) {
    if( s[i] == '\r')
        t[i] = '\n';
    else
        t[i] = s[i];
}
t[i] = 0;
return strdup( s);
}

```

#### 7.5.4.3 void hex\_dump( int n, unsigned char \* s )

Prints a hex dump of the given data.

Used to debug packet data.

##### Parameters

in	n	Number of bytes passed in s
in	s	Data to dump

Definition at line 529 of file lspmac.c.

```

{
int i;           // row counter
int j;           // column counter
unsigned char outs[128], outs1[4];

for( i=0; n > 0; i++ ) {

    sprintf( (char *)outs, "%04d: ", 16*i);
    for( j=0; j<16 && n > 0; j++ ) {
        if( j==8)
            strcat( (char *)outs, " ");
        sprintf( (char *)outs1, " %02x", *(s + 16*i + j));
        strcat( (char *)outs, (char *)outs1);
        n--;
    }
    lslogging_log_message( "hex_dump: %s", outs);
}
}

```

#### 7.5.4.4 void lsConnect ( char \* ipaddr )

Connect to the PMAC socket.

Establish or reestablish communications.

##### Parameters

in	ipaddr	String representation of the IP address (dot quad or FQN)
----	--------	---

Definition at line 577 of file lspmac.c.

```

{
int psock;           // our socket: value stored in pmacfd.fd
int err;             // error code from some system calls
struct sockaddr_in *addrP; // our address structure to connect to
struct addrinfo ai_hints; // required for getaddrinfo
struct addrinfo *ai_resultP; // linked list of address structures (we'll
                           // always pick the first)

pmacfd.fd      = -1;
pmacfd.events = 0;

// Initial buffer(s)
memset( &ai_hints, 0, sizeof( ai_hints));

ai_hints.ai_family   = AF_INET;
ai_hints.ai_socktype = SOCK_STREAM;

//
// get address
//
err = getaddrinfo( ipaddr, NULL, &ai_hints, &ai_resultP);
if( err != 0) {

    lslogging_log_message( "Could not find address: %s",
                           gai_strerror( err));

    return;
}

addrP = (struct sockaddr_in *)ai_resultP->ai_addr;
addrP->sin_port = htons( PMACPORT);

psock = socket( PF_INET, SOCK_STREAM, 0);
if( psock == -1) {
    lslogging_log_message( "Could not create socket");
    return;
}

err = connect( psock, (const struct sockaddr *)addrP, sizeof( *addrP));
if( err != 0) {
    lslogging_log_message( "Could not connect socket: %s",
                           strerror( errno));
    return;
}

```

```

ls_pmac_state = LS_PMAC_STATE_IDLE;
pmacfd.fd      = psoc;
pmacfd.events  = POLLIN;

}

```

#### 7.5.4.5 void lspmac\_abort( )

abort motion and try to recover

Definition at line 2104 of file lspmac.c.

```

{
// Stop everything! (consider ^O instead of ^A)
// lspmacc_SockSendDPControlChar( "Abort Request", 0
x01);

//
// and reset motion flag
//
lspmacc_SockSendDPLine( "Reset", "%s", "M5075=0");

}

```

#### 7.5.4.6 void lspmac\_asciiCmdCB ( pmac\_cmd\_queue\_t \* cmd, int nreceived, char \* buf )

PMAC has received our ascii command request Now see when it is ready for the next one.

Definition at line 2018 of file lspmac.c.

```

lspmacc_get_ascii( cmd->event );
{

```

#### 7.5.4.7 void lspmac\_backLight\_down\_cb ( char \* event )

Turn off the backlight whenever it goes down.

##### Parameters

<code>event</code>	Name of the event that called us
--------------------	----------------------------------

Definition at line 4058 of file lspmac.c.

```

blight->moveAbs( blight, 0.0);
{

```

#### 7.5.4.8 void lspmac\_backLight\_up\_cb ( char \* event )

Turn on the backlight whenever it goes up.

##### Parameters

<code>event</code>	Name of the event that called us
--------------------	----------------------------------

Definition at line 4051 of file lspmac.c.

```

    blight->moveAbs( blight, (int)(lspmac_getPosition
        ( zoom)));
}

```

#### 7.5.4.9 `lspmac.bi_t* lspmac.bi_init( lspmac.bi_t * d, int * ptr, int mask, char * onEvent, char * offEvent )`

Initialize binary input.

Definition at line 3761 of file `lspmac.c`.

```

{
    lspmac_nbis++;
    pthread_mutex_init( &(d->mutex), NULL);
    d->ptr      = ptr;
    d->mask     = mask;
    d->changeEventOn = strdup( onEvent);
    d->changeEventOff = strdup( offEvent);
    d->first_time = 1;

    lsevents_preregister_event( "%s", d->changeEventOn
        );
    lsevents_preregister_event( "%s", d->changeEventOff
        );

    return d;
}

```

#### 7.5.4.10 `void lspmac.blight_lut_setup( )`

Set up lookup table for blight.

Definition at line 4251 of file `lspmac.c`.

```

{
    int i;
    lsredis_obj_t *p;

    pthread_mutex_lock( &blight->mutex);

    blight->nlut = 11;
    blight->lut = calloc( 2 * blight->nlut, sizeof( double));
    if( blight->lut == NULL) {
        lslogging_log_message( "lspmac_blight_lut_setup: out
            of memory");
        exit( -1);
    }

    blight->lut[0] = 0;
    blight->lut[1] = 0;

    for( i=1; i<blight->nlut; i++) {
        p = lsredis_get_obj( "cam.zoom.%d.LightIntensity", i);
        if( p==NULL || strlen( lsredis_getstr(p)) == 0) {
            free( blight->lut);
            blight->lut = NULL;
            blight->nlut = 0;
            pthread_mutex_unlock( &blight->mutex);
            lslogging_log_message( "lspmac_blight_lut_setup:
                cannot find MotorPosition element for cam.bright level %d", i);
            return;
        }
        blight->lut[2*i] = i;
        blight->lut[2*i+1] = 20000.0 * lsredis_getd( p) / 100.
            0;
    }
    for( i=0; i<blight->nlut; i++) {
        lslogging_log_message( "lspmac_blight_lut_setup: i:
            %d x: %f y: %f y(lut): %f x(rlut): %f",
            i, blight->lut[2*i], blight->lut[2
            *i+1],
            lspmac_lut( blight->nlut, blight
            ->lut, blight->lut[2*i]),
            lspmac_rlut( blight->nlut,
            blight->lut, blight->lut[2*i+1])
    }
}

```

```

    );
pthread_mutex_unlock( &blight->mutex);
}

```

#### 7.5.4.11 `lspmac_motor_t* lspmac_bo_init( lspmac_motor_t * d, char * name, char * write_fmt, int * read_ptr, int read_mask )`

Initialize binary i/o motor.

##### Parameters

in	<i>d</i>	Our uninitialized motor object
in	<i>name</i>	Name of motor to coordinate with DB
in	<i>write_fmt</i>	Format string used to generate PMAC command to move motor
in	<i>read_ptr</i>	Pointer to byte in md2_status to find position
in	<i>read_mask</i>	Bitmask to find position in *read_ptr

Definition at line 3690 of file lspmac.c.

```

{
    _lspmac_motor_init( d, name);

    d->moveAbs      = lspmac_moveabs_bo_queue;
    d->jogAbs       = lspmac_moveabs_bo_queue;
    d->read          = lspmac_bo_read;
    d->write_fmt     = strdup( write_fmt);
    d->read_ptr      = read_ptr;
    d->read_mask     = read_mask;

    lsevents_preregister_event( "%s 1", d->name);
    lsevents_preregister_event( "%s 0", d->name);
    return d;
}

```

#### 7.5.4.12 `void lspmac_bo_read( lspmac_motor_t * mp )`

Read the state of a binary i/o motor This is the read method for the binary i/o motor class.

##### Parameters

in	<i>mp</i>	The motor
----	-----------	-----------

Definition at line 1155 of file lspmac.c.

```

{
int pos, changed;
char *fmt;

pthread_mutex_lock( &(mp->mutex) );

pos = (*mp->read_ptr) & mp->read_mask == 0 ? 0 : 1;

changed = pos != mp->position;
mp->position = pos;

if( changed) {
    mp->motion_seen = 1;
    mp->not_done = 0;
    mp->command_sent = 1;
    pthread_cond_signal( &(mp->cond));
    lsevents_send_event( "%s Moving", mp->name);
    lsevents_send_event( "%s %d", mp->name, pos);
    lsevents_send_event( "%s In Position", mp->name);
}

if( mp->reported_position != mp->position) {
    fmt = lsredis_getstr(mp->redis_fmt);
}

```

```

lsredis_setstr( mp->redis_position, fmt, mp->
    position);
free(fmt);
mp->reported_position = mp->position;
}

pthread_mutex_unlock( &(mp->mutex));
}

```

#### 7.5.4.13 void lspmac\_command\_done\_cb ( char \* event )

Definition at line 4328 of file lspmac.c.

```

{
int i;
char s[32];
lspmac_motor_t *mp;

s[0] = 0;
for( i=0; i<sizeof(s)-1 && event[i]; i++ ) {
    s[i] = 0;
    if( event[i] == ' ' )
        break;
    s[i] = event[i];
}

mp = lspmac_find_motor_by_name( s );

if( mp == NULL)
    return;

pthread_mutex_lock( &(mp->mutex));

mp->command_sent = 1;

pthread_cond_signal( &(mp->cond));
pthread_mutex_unlock( &(mp->mutex));

return;
}

```

#### 7.5.4.14 void lspmac\_cryoSwitchChanged\_cb ( char \* event )

Definition at line 3981 of file lspmac.c.

```

{
int pos;

pthread_mutex_lock( &(cryo->mutex));
pos = cryo->position;
pthread_mutex_unlock( &(cryo->mutex));

cryo->moveAbs( cryo, pos ? 0.0 : 1.0);
}

```

#### 7.5.4.15 lspmac\_motor\_t\* lspmac\_dac\_init ( lspmac\_motor\_t \* d, int \* posp, char \* mvar, char \* name, int(\*)(lspmac\_motor\_t \*,double) moveAbs )

Initialize DAC motor Note that some motors require further initialization from a database query.

For this reason this initialization code must be run before the database queue is allowed to be processed.

##### Parameters

<i>out</i>	<i>d</i>	Returns the (almost) initialized motor object [in,out] unitintialized motor
<i>in</i>	<i>posp</i>	Location of current position
<i>in</i>	<i>mvar</i>	M variable, ie, "M1200"
<i>in</i>	<i>name</i>	name to coordinate with DB
<i>in</i>	<i>moveAbs</i>	Method to use to move this motor

Definition at line 3719 of file lspmacc.c.

```
{
    lspmacc_motor_init( d, name);
    d->moveAbs      = moveAbs;
    d->jogAbs       = moveAbs;
    d->read          = lspmacc_dac_read;
    d->actual_pos_ctns_p = posp;
    d->dac_mvar     = strdup(mvar);

    return d;
}
```

#### 7.5.4.16 void lspmacc\_dac\_read ( lspmacc\_motor\_t \* mp )

Read a DAC motor position.

##### Parameters

in	mp	The motor
----	----	-----------

Definition at line 1190 of file lspmacc.c.

```
{
double u2c;
char *fmt;

pthread_mutex_lock( &(mp->mutex));
mp->actual_pos_ctns = *mp->actual_pos_ctns_p;
u2c = lsredis_getd( mp->u2c);

if( mp->nlut >0 && mp->lut != NULL) {
    if( u2c == 0.0)
        u2c = 1.0;
    mp->position = lspmacc_rlut( mp->nlut, mp->lut, mp
        ->actual_pos_ctns/u2c);
} else {
    if( u2c != 0.0) {
        mp->position = mp->actual_pos_ctns / u2c;
    } else {
        mp->position = mp->actual_pos_ctns;
    }
}

if( fabs(mp->reported_position - mp->position) >=
    lsredis_getd(mp->update_resolution)) {
    fmt = lsredis_getstr(mp->redis_fmt);
    lsredis_setstr( mp->redis_position, fmt, mp->
        position);
    free(fmt);
    mp->reported_position = mp->position;
}

pthread_mutex_unlock( &(mp->mutex));
}
```

#### 7.5.4.17 void lspmacc\_Error ( char \* buff )

The service routing detected an error condition.

Scan the response buffer for an error code and print it out.

##### Parameters

in	buff	Buffer returned by PMAC perhaps containing a NULL terminated message.
----	------	---

Definition at line 786 of file lspmacc.c.

```
{
```

```

int err;
//
// assume buff points to a 1400 byte array of stuff read from the pmac
//

if( buff[0] == 7 && buff[1] == 'E' && buff[2] == 'R' && buff[3] == 'R' ) {
    buff[7] = 0; // For null termination
    err = atoi( &(buff[4]) );
    if( err > 0 && err < 20 ) {
        lslogging_log_message( pmac_error_strs
            [err]);
    }
}
lspmac_Reset();
}

```

#### 7.5.4.18 int lspmac\_est\_move\_time ( double \* est\_time, int \* mmaskp, lspmac\_motor\_t \* mp\_1, int jog\_1, char \* preset\_1, double end\_point\_1, ... )

Move the motors and estimate the time it'll take to finish the job.

Returns the estimate time and the coordinate system mask to wait for

##### Parameters

<i>est_time</i>	Returns number of seconds we estimate the move(s) will take
<i>mmaskp</i>	Mask of coordinate systems we are trying to move, excluding jogs. Used to wait for motions to complete
<i>mp_1</i>	Pointer to first motor
<i>jog_1</i>	1 to force a jog, 0 to try a motion program DO NOT MIX JOGS AND MOTION PROGRAMS IN THE SAME COORDINATE SYSTEM!
<i>preset_1</i>	Name of preset we'd like to move to or NULL if <i>end_point_1</i> should be used instead
<i>end_point_1</i>	End point for the first motor. Ignored if <i>preset_1</i> is non null and identifies a valid preset for this motor
...	Perhaps more quads of motors, jog flags, preset names, and end points. End is a NULL motor pointer MUST END ARG LIST WITH NULL

< units to counts  
< The total distance we need to go  
< Our maximum velocity  
< Our maximum acceleration  
< Total time for this motor  
< coordinate system motion flags

Definition at line 2829 of file lspmac.c.

```

{
static char axes[] = "XYZUVWABC";
int qs[9];
lspmac_combined_move_t motions[32];
char s[256];
int foundone;
int moving_flags;
struct timespec timeout;
int j;
va_list arg_ptr;
lspmac_motor_t *mp;
double ep, maybe_ep;
char *ps;
double
min_pos,
max_pos,
neutral_pos,
u2c,
D,
V,

```

```

A,
Tt;
int err;
int jog;
int i;
uint32_t m5075;

// reset our coordinate flags and command strings
//
for( i=0; i<32; i++) {
    motions[i].moveMe = 0;
}
m5075 = 0;
if( mmaskp != NULL)
    *mmaskp = 0;

//
// Initialize first iteration
//
*est_time = 0.0;
mp = mp_1;
ps = preset_1;
ep = end_point_1;
jog = jog_1;

va_start( arg_ptr, end_point_1);
while( 1) {
/*
 *      :           | Constant          |
 *      :           | <-- Velocity   ---> |
 *      :           |           Time (Ct)   |
 * V :-----+
 * e :           / \-----+
 * l :           / \-----+
 * o :           / \-----+
 * c :           / \-----+
 * i :           / \-----+
 * t :           / \-----+
 * y :-----/ \-----+
 *          | |           v-----+
 *          | |           Time
 *          -->| | <-- Acceleration Time (At)
 *          | |
 *          | <---- Total Time (Tt) ----->|
 *
 *      Assumption 1: We can replace S curve acceleration with linear
 *      acceleration
 *      for the purposes of distance and time calculations for the timeout
 *      period that we are attempting to calculate here.
 *
 *      Ct = Constant Velocity Time. The time spent at constant velocity.
 *
 *      At = Acceleration Time. Time spent accelerating at either end of
 *      the ramp, that is,
 *      1/2 the total time spent accelerating and decelerating.
 *
 *      D = the total distance we need to travel
 *
 *      V = constant velocity. Here we use the motor's maximum velocity.
 *
 *      A = the motor acceleration, Here it's the maximum acceleration.
 *
 *      V = A * At
 *
 *      or At = V/A
 *
 *      The Total Time (Tt) is
 *
 *      Tt = Ct + 2 * At
 *
 *
 *      If we had infinite acceleration the total time would be D/V. To
 *      account for finite acceleration we just need to
 *      adjust this for the average velocity while accelerating (0.5 V).
 *      This neatly adds a single V/A term:
 *
 *      (1)      Tt = D/V + V/A
 *
 *      When the distance is short, we need a different calculation:
 *
 *      D = 0.5 * A * T1^2 + 0.5 * A * T2^2 (T1 = acceleration time and
 *      T2 = deceleration time)
 *
 *      or, since total time Tt = T1 + T2 and T1 = T2,
 */
}

```

```

*
*      D = A * (0.5*Tt)^2
*
*      or
*
*      (2)      Tt = 2 * sqrt( D/A)
*
*
*      When we accelerate to the maximum speed the time it takes is V/A so
*      the distance we travel (Da) is
*
*      Da = 0.5 * A * (V/A)^2
*
*      or
*
*      Da = 0.5 * V^2 / A
*
*      So when D > 2 * Da, or
*
*      D > V^2 / A
*
*      we need to use equation (1) otherwise we need to use equation (2)
*
*/
}

if( mp->magic != LSPMAC_MAGIC_NUMBER) {
    lslogging_log_message( "lspmac_est_move_time:
        WARNING: bad motor structure. Check that your motor list is NULL terminated.");
    break;
}

lslogging_log_message( "lspmac_est_move_time: find
    motor %s, jog %d, preset %s, endpoint %f",
    mp->name, jog, ps == NULL ? "NULL" : ps, ep);

Tt = 0.0;
if( mp != NULL && mp->max_speed != NULL && mp->max_accel
!= NULL && mp->u2c != NULL) {

    //
    // get the real endpoint if a preset was mentioned
    //
    if( ps != NULL && *ps != 0) {
        err = lsredis_find_preset( mp->name, ps, &
        maybe_ep);
        if( err != 0)
            ep = maybe_ep;
    }

    u2c = lsredis_getd( mp->u2c);

    //
    // For look up tables user units are (or should be) counts and u2c should
    be 1
    //
    if( mp->nlut > 0 && mp->lut != NULL) {
        u2c = 1.0;
        D = lspmac_lut( mp->nlut, mp->lut, ep) - lspmac_lut
        ( mp->nlut, mp->lut, lspmac_getPosition( mp));
    } else {
        D = ep - lspmac_getPosition( mp);
        // User units
    }
}

V = lsredis_getd( mp->max_speed) / u2c * 1000.;
// User units per second
A = lsredis_getd( mp->max_accel) / u2c * 1000. *
1000;           // User units per second per second

neutral_pos = lsredis_getd( mp->neutral_pos);
min_pos     = lsredis_getd( mp->min_pos) - neutral_pos
;
max_pos     = lsredis_getd( mp->max_pos) - neutral_pos
;

if( ep < min_pos || ep > max_pos) {
    lslogging_log_message( "lspmac_est_move_time:
        Motor %s Requested position %f out of range: min=%f, max=%f", mp->name, ep,
        min_pos, max_pos);
    lsevents_send_event( "%s Move Aborted", mp->name
);
    return 1;
}

mp->requested_position = ep;
mp->requested_pos_cnts = u2c * (mp->requested_position

```

```

+ neutral_pos);

// Don't bother with motors without velocity or acceleration defined
//
if( V > 0.0 && A > 0.0) {
    if( fabs(D) > V*V/A) {
        //
        // Normal ramp up, constant velocity, and ramp down
        //
        Tt = fabs(D)/V + V/A;
    } else {
        //
        // Never reach constant velocity, just ramp up a bit and back down
        //
        Tt = 2.0 * sqrt( fabs(D)/A);
    }

    lslogging_log_message( "lspmac_est_move_time:
Motor: %s D: %f VV/A: %f Tt: %f", mp->name, D, V*V/A, Tt);
} else {
    //
    // TODO: insert move time based for DAC or BO motor like objects;
    // For now assume 100 msec;
    //
    Tt = 0.1;
}

// Perhaps flag a coordinate system
//
// We can move a motor that's not in a coordinate system but we cannot
move a motor that is but does not
// have an axis defined if we are also moving one that does. It's a
limitation, I guess.
//
if( jog != 1 &&
    mp->coord_num != NULL && lsredis_getl( mp->
coord_num) > 0 && lsredis_getl( mp->coord_num) <=
16 &&
    mp->motor_num != NULL && lsredis_getl( mp->
motor_num) > 0 && mp->axis != NULL && lsredis_getc( mp
->axis) != 0) {
    int axis;
    int motor_num;

    motor_num = lsredis_getl( mp->motor_num);

    axis = lsredis_getc( mp->axis);
    for( j=0; j<sizeof(axes); j++) {
        if( axis == axes[j])
            break;
    }

    if( j < sizeof( axes)) {
        //
        // Store the motion request for a normal PMAC motor
        //
        int cn;
        int in_position_band;

        cn = lsredis_getl( mp->coord_num);
        in_position_band = lsredis_getl( mp->in_position_band
    );
}

motions[motor_num - 1].coord_num = cn;
motions[motor_num - 1].axis      = j;
motions[motor_num - 1].Delta     = D * u2c;
//
// Don't ask to run a motion program if we are already where we want
to be
//
// Deadband is 10 counts except for zoom which is 100.
// We use Ixx28 In-Position Band which has units of 1/16 count
//
if( abs(motions[motor_num - 1].Delta)*16 >= in_position_band) {
    m5075 |= (1 << (cn - 1));
    motions[motor_num - 1].moveme   = 1;
}
lslogging_log_message( "lspmac_est_move_time:
moveme=%d motor '%s' motions index=%d coord_num=%d axis=%d Delta=%d    m5075=%u",
motions[motor_num-1].moveme, mp->name,
motor_num -1, motions[motor_num-1].coord_num, motions[motor_num-1].axis
, motions[motor_num-1].Delta,
m5075);
}

```

```

} else {
//
// Here we are dealing with a DAC or BO motor or just want to jog.
//
if( mp->jogAbs( mp, ep) ) {
    lslogging_log_message( "lspmac_est_move_time:
motor %s failed to queue move of distance %f from %f", mp->name, D,
lspmac_getPosition(mp));
    lsevents_send_event( "Move Aborted");
    return 1;
}
//
// Update the estimated time
//
*est_time = *est_time < Tt ? Tt : *est_time;

lslogging_log_message( "lspmac_est_move_time:
est_time=%f", *est_time);

}

mp = va_arg( arg_ptr, lspmac_motor_t * );
if( mp == NULL)
    break;

jog = va_arg( arg_ptr, int);
ps  = va_arg( arg_ptr, char *);
ep  = va_arg( arg_ptr, double);

}
va_end( arg_ptr);

// Set the motion program flags
//
if( m5075 != 0) {
    if( mmaskp != NULL)
        *mmaskp |= m5075; // Tell the caller about our new mask

pthread_mutex_lock( &lspmac_moving_mutex);
moving_flags = lspmac_moving_flags;
pthread_mutex_unlock( &lspmac_moving_mutex);

if( (moving_flags & m5075) != m5075) {
    lspmac_SockSendDPLine( NULL, "M5075=(M5075 | %d)",
m5075);

pthread_mutex_lock( &lspmac_moving_mutex);
clock_gettime( CLOCK_REALTIME, &timeout);
//
timeout.tv_sec += 2;      // 2 seconds should be more than enough time to
set the flags
err = 0;
while( err == 0 && ((lspmac_moving_flags & m5075) !=
m5075))
    err = pthread_cond_timedwait( &lspmac_moving_cond, &
lspmac_moving_mutex, &timeout);
moving_flags = lspmac_moving_flags;
pthread_mutex_unlock( &lspmac_moving_mutex);

if( ((moving_flags & m5075) != m5075) && err == ETIMEDOUT) {
    lslogging_log_message( "lspmac_est_move_time:
Timed out waiting for moving flags. lspmac_moving_flags = 0x%0x, looking for 0x%0x
test exp: 0x%0x test: %d",
moving_flags, m5075, (moving_flags & m5075), (
moving_flags & m5075) != m5075);
    lsevents_send_event( "Combined Move Aborted");
    return 1;
}
}

for( i=1; i<=16; i++) {
//
// Loop over coordinate systems
//
foundone = 0;

for( j=0; j<9; j++)
    qs[j] = 0;

for( j=0; j<31; j++) {

```

```

// Loop over motors
//
if( motions[j].moveme && motions[j].coord_num == i) {
    if( abs(motions[j].Delta) > 0) {
        qs[(int)(motions[j].axis)] = motions[j].Delta;
        foundone=1;
    }
}
}

if( foundone) {
    sprintf( s, "&%d Q40=%d Q41=%d Q42=%d Q43=%d Q44=%d Q45=%d Q46=%d Q47=%d
    Q48=%d Q49=%d Q100=%d B180R",
    i, qs[0], qs[1], qs[2], qs[3], qs[4], qs[5], qs[6], qs[7], qs[8]
    , *est_time * 1000., 1 << (i-1));
    lspmac_SockSendDPLine( NULL, s);
}

return 0;
}

```

#### 7.5.4.19 int lspmac\_est\_move\_time\_wait ( double move\_time, int cmask, lspmac\_motor\_t \*mp\_1, ... )

wait for motion to stop returns non-zero if the wait timed out

##### Parameters

<i>move_time</i>	The time out in seconds
<i>cmask</i>	A coordinate system mask to wait for
<i>mp_1</i>	NULL terminated list of individual motors to wait for

Both values are returned from lspmac\_est\_move\_time

Definition at line 3184 of file lspmac.c.

```

{
int err;
double isecs, fsecs;
struct timespec timeout;
va_list arg_ptr;
lspmac_motor_t *mp;

clock_gettime( CLOCK_REALTIME, &timeout);
fsecs = modf( move_time, &isecs);
timeout.tv_sec += (long)floor(isecs);
timeout.tv_nsec += (long)floor(fsecs * 1.e9);
timeout.tv_sec += timeout.tv_nsec / 1000000000;
timeout.tv_nsec %= 1000000000;

err = 0;
pthread_mutex_lock( &lspmac_moving_mutex);
while( err == 0 && (lspmac_moving_flags & cmask) != 0)
    err = pthread_cond_timedwait( &lspmac_moving_cond, &
        lspmac_moving_mutex, &timeout);
pthread_mutex_unlock( &lspmac_moving_mutex);

if( err != 0) {
    if( err == ETIMEDOUT) {
        lslogging_log_message( "
        ltest_lspmac_est_move_time_wait: timed out waiting %f seconds, cmask = 0x%0x", move_time, cmask);
    }
    lspmac_abort();
    return 1;
}

va_start( arg_ptr, mp_1);
for( mp = mp_1; mp != NULL; mp = va_arg( arg_ptr, lspmac_motor_t
*)) {
    if( mp->magic != LSPMAC_MAGIC_NUMBER) {
        lslogging_log_message( "lspmac_est_move_time_wait:
        WARNING: motor list must be NULL terminated. Check your call to
        lspmac_est_move_time_wait.");
    }
}

```

```

    if( lspmac_moveabs_wait( mp, move_time) ) {
        lslogging_log_message( "lspmac_est_move_time_wait:
            timed out waiting %f seconds for motor %s", move_time, mp->name);
        return 1;
    }
}
va_end( arg_ptr);

return 0;
}

```

#### 7.5.4.20 `lspmac_motor_t* lspmac_find_motor_by_name( char * name )`

Definition at line 4311 of file lspmac.c.

```

{
lspmac_motor_t *rtn;
ENTRY entry_in, *entry_outp;
int err;

entry_in.key = name;
entry_in.data = NULL;
err = hsearch_r( entry_in, FIND, &entry_outp, &motors_ht );
if( err == 0 ) {
    lslogging_log_message( "lspmac_find_motor_by_name:
        hsearch_r failed for motor '%s': %s", name, strerror( errno));
    return NULL;
}
rtn = entry_outp->data;

return rtn;
}

```

#### 7.5.4.21 `void lspmac_flight_lut_setup( )`

Set up lookup table for flight.

Definition at line 4218 of file lspmac.c.

```

{
int i;
lsredis_obj_t *p;

pthread_mutex_lock( &flight->mutex );

flight->nlut = 11;
flight->lut = calloc( 2 * flight->nlut, sizeof( double));
if( flight->lut == NULL) {
    lslogging_log_message( "lspmac_flight_lut_setup: out
        of memory");
    exit( -1);
}

flight->lut[0] = 0;
flight->lut[1] = 0;
for( i=1; i < flight->nlut; i++) {
    p = lsredis_get_obj( "cam.zoom.%d.FrontLightIntensity", i);
    if( p==NULL || strlen( lsredis_getstr(p)) == 0) {
        free( flight->lut);
        flight->lut = NULL;
        flight->nlut = 0;
        pthread_mutex_unlock( &flight->mutex);
        lslogging_log_message( "lspmac_flight_lut_setup:
            cannot find MotorPosition element for cam.light level %d", i);
        return;
    }
    flight->lut[2*i] = i;
    flight->lut[2*i+1] = 32767.0 * lsredis_getd( p) / 100.
        0;
}
pthread_mutex_unlock( &flight->mutex);
}

```

## 7.5.4.22 void lspmac\_fscint\_lut\_setup( )

Set up lookup table for fscint.

Definition at line 4292 of file lspmac.c.

```

{
int i;

pthread_mutex_lock( &fscint->mutex);

fscint->nlut = 101;
fscint->lut = calloc( 2 * fscint->nlut, sizeof( double));
if( fscint->lut == NULL) {
    lslogging_log_message( "lspmac_fscint_lut_setup: out
        of memory");
    exit( -1);
}

for( i=0; i<fscint->nlut; i++) {
    fscint->lut[2*i] = i;
    fscint->lut[2*i+1] = 320.0 * i;
}
pthread_mutex_unlock( &fscint->mutex);
}
```

## 7.5.4.23 lspmac\_motor\_t\* lspmac\_fshut\_init( lspmac\_motor\_t \* d )

Initialize the fast shutter motor.

## Parameters

in	d	Our uninitialized motor object
----	---	--------------------------------

Definition at line 3673 of file lspmac.c.

```

{
_lspmac_motor_init( d, "fastShutter");

d->moveAbs      = lspmac_moveabs_fshut_queue
;
d->jogAbs       = lspmac_moveabs_fshut_queue
;
d->read          = lspmac_shutter_read;

return d;
}
```

## 7.5.4.24 void lspmac\_full\_card\_reset\_cb( char \* event )

reset and restart

Definition at line 3778 of file lspmac.c.

```

{
lspmac_running = 0;
pthread_join( pmac_thread, NULL);
pthread_mutex_lock( &pmac_queue_mutex);

ethCmdOn     = 0;
ethCmdOff    = 0;
ethCmdReply  = 0;

lspmac_running = 1;
ls_pmac_state = LS_PMAC_STATE_DETACHED;

pthread_mutex_unlock( &pmac_queue_mutex);

lspmac_init( 0, 0);
lspmac_run();
}
```

#### 7.5.4.25 void lspmacc\_get\_ascii( char \* event )

Forward declaration.

Request the ascii buffers from the PMAC.

Definition at line 2010 of file lspmacc.c.

```

    {
lspmacc_send_command( VR_UPLOAD, VR_PMAC_GETMEM
    , 0x0e9c, 0, sizeof(lspmacc_ascii_buffers_t), NULL,
    lspmacc_get_ascii_cb, 0, event);
}
}
```

#### 7.5.4.26 void lspmacc\_get\_ascii\_cb( pmac\_cmd\_queue\_t \* cmd, int nreceived, char \* buff )

service the ascii buffer request response

Definition at line 1896 of file lspmacc.c.

```

{
uint32_t clrdata;
int need_more;

need_more = 0;
pthread_mutex_lock( &lspmacc_ascii_mutex );
memcpy( &lspmacc_ascii_buffers, buff, sizeof(
    lspmacc_ascii_buffers));

// 
// The response is not ready yet
// This will be an infinite loop if we queue a command that does not
// produce a response.
//
// Quoted comments below from Delta Tau "Turbo PMAC User Manual 9/12/2008,
// page 422"
//
// "1. Wait for the Host-Input Control Word at 0x0F40 (Y:$063D0) to become
// greater than 0, indicating
// that a response line is ready."
//
if( lspmacc_ascii_buffers.response_buf == 0) {
    need_more = 1;
} else {
    if( (lspmacc_ascii_buffers.response_buf & 0
        x8000) != 0) {
        char bcd1, bcd2, bcd3;
        int errcode;
        // Error response
        //
        // "2. Interpret the value in this register to determine what
        // type of response is present. If Bit 15 is 1, Turbo PMAC is
        // reporting an error in the command, and there is no response
        // other than this word. In this case, Bits 0 - 11 encode the
        // error number for the command as 3 BCD digits."
        //
        need_more = 0;
        bcd1 = lspmacc_ascii_buffers.response_buf
            & 0x000f;
        bcd2 = (lspmacc_ascii_buffers.response_buf
            & 0x00f0) >> 4;
        bcd3 = (lspmacc_ascii_buffers.response_buf
            & 0x0f00) >> 8;
        errcode = (bcd3 * 10 + bcd2) * 10 + bcd1;

        if( errcode >= sizeof( pmac_error_strs)/sizeof(
            *pmac_error_strs))
            errcode = 0;
        lslogging_log_message( "lspmacc_get_ascii_cb: Error
            returned for %s: %s", lspmacc_ascii_buffers.command_str
            , pmac_error_strs[errcode]);
        //
        // Command not allowed during program execution.
        //
        // Requeue it;
        if( errcode == 1) {
            lspmacc_dpascii_off--;
        }
    }
}
}
```

```

} else {
//
// "3. Read the response string starting at 0x0F44
// (Y:$0603D1). Two 8-bit characters are packed into each 16-bit
// word; the first character is placed into the low
// byte. Subsequent characters are placed into consecutive
// higher addresses, two per 16-bit word. (In byte addressing,
// each character is read from an address one higher than the
// preceding character.) Up to 255 characters can be sent in a
// single response line. The string is terminated with the NULL
// character (byte value 0), convenient for C-style string
// handling. For Pascal-style string handling, the register at
// 0x0F42 (X:$0603D0) contains the number of characters in the
// string (plus one)."
//
if( cmd->event != NULL && strncmp( cmd->event, "Control-", 8)
== 0) {
    lslogging_log_message( "%s: %s", cmd->event,
lspmac_ascii_buffers.response_str);
    need_more = 0;
} else {
    if( lspmac_ascii_buffers.response_n > 1)
        lslogging_log_message( "lspmac_get_ascii_cb:
'%s' '%s'", lspmac_ascii_buffers.command_str,
lspmac_ascii_buffers.response_str);
    else
        lslogging_log_message( "lspmac_get_ascii_cb:
'%s' responded", lspmac_ascii_buffers.command_str)
;
}

//
// 5. If Bits 0 - 7 of the Host-Input Control Word had
// contained the value $0D (13 decimal, "CR"), this was not the
// last line in the response, and steps 1 - 4 should be
// repeated. If they had contained the value $06 (6 decimal,
// "ACK"), this was the last line in the response."
//
if( (lspmac_ascii_buffers.response_buf
& 0x00ff) == 0x0d) {
    need_more = 1;
} else {
    need_more = 0;

    if( cmd->event != NULL && *(cmd->event) != 0)
        lsevents_send_event( "%s command accepted", cmd
->event);
}
}
}

pthread_mutex_unlock( &lspmac_ascii_mutex);

//
// Reset the buffer flags and, perhaps, requeue a request
//
// "4. Clear the Host-Input Control Word at 0x0F40 (Y:$063D0)
// to 0. Turbo PMAC will not send another response line until it sees
// this register set to 0."
//
clrdta = 0;           // set the control word to zero

if( need_more) {
    lspmac_send_command( VR_UPLOAD, VR_PMAC_SETMEM
    , 0x0f40, 0, 4, (char *)&clrdta, lspmac_more_ascii_cb, 1,
    NULL);
} else {
    lspmac_send_command( VR_UPLOAD, VR_PMAC_SETMEM
    , 0x0f40, 0, 4, (char *)&clrdta, NULI, 1, NULL);
    lspmac_ascii_busy = 0;
}
}
}

```

#### 7.5.4.27 void lspmacc\_get\_status( )

Request a status update from the PMAC.

Definition at line 1884 of file lspmacc.c.

```

{
lspmac_send_command( VR_UPLOAD, VR_PMAC_GETMEM

```

```

    , 0x400, 0, sizeof(md2_status_t), NULL, lspmac_get_status_cb
    , 0, NULL);
}

```

#### 7.5.4.28 void lspmac\_get\_status\_cb ( pmac\_cmd\_queue\_t \* cmd, int nreceived, char \* buff )

Service routing for status update This updates positions and status information.

##### Parameters

in	<i>cmd</i>	The command that generated this reply
in	<i>nreceived</i>	Number of bytes received
in	<i>buff</i>	The Big Byte Buffer

Definition at line 1661 of file lspmac.c.

```

{
#ifndef SHOW_RATE
static struct timespec ts1;
static struct timespec ts2;
static int cnt = 0;
#endif

int i;
lspmac_bi_t *bp;

clock_gettime( CLOCK_REALTIME, &lspmac_status_time);

#ifndef SHOW_RATE
if( cnt == 0) {
    clock_gettime( CLOCK_REALTIME, &ts1);
}
#endif

pthread_mutex_lock( &md2_status_mutex);
memcpy( &md2_status, buff, sizeof(md2_status));
// 
// Note that we are the only thread that writes to md2_status
// so we no longer need the lock to read. Other threads must
// lock the mutex to read md2_status.
//
pthread_mutex_unlock( &md2_status_mutex);

//
// track the coordinate system moving flags
//
pthread_mutex_lock( &lspmac_moving_mutex);
if( md2_status.moving_flags != lspmac_moving_flags
    ) {
    int mask;

    lslogging_log_message( "lspmac_get_status_cb: new
        moving flag: %0x", md2_status.moving_flags);
    mask = 1;
    for( i=1; i<=16; i++, mask <<= 1) {
        if( ((lspmac_moving_flags & mask) != 0) && ((

        md2_status.moving_flags & mask) == 0)) {
            // Falling edge: send event
            lsevents_send_event( "Coordsys %d Stopped", i);
        }
    }
    lspmac_moving_flags = md2_status.moving_flags
    ;
    pthread_cond_signal( &lspmac_moving_cond);
}

pthread_mutex_unlock( &lspmac_moving_mutex);

//
// Read the motor positions
//
for( i=0; i<lspmac_nmotors; i++) {
    lspmac_motors[i].read(&(lspmac_motors[i]));
}

//
// Read the binary inputs and perhaps send an event

```

```

//  

for( i=0; i<lspmac_nbis; i++) {  

    bp = &(lspmac_bis[i]);  

  

    pthread_mutex_lock( &(bp->mutex));  

  

    bp->position = (*bp->ptr) & bp->mask) == 0 ? 0 : 1;  

  

    if( bp->first_time) {  

        bp->first_time = 0;  

        if( bp->position==1 && bp->changeEventOn != NULL &&  

            bp->changeEventOn[0] != 0)  

            lsevents_send_event( lspmac_bis[i].  

                changeEventOn);  

        if( bp->position==0 && bp->changeEventOff != NULL  

            && bp->changeEventOff[0] != 0)  

            lsevents_send_event( lspmac_bis[i].  

                changeEventOff);  

    } else {  

        if( bp->position != bp->previous) {  

            if( bp->position==1 && bp->changeEventOn != NULL  

                && bp->changeEventOn[0] != 0)  

                lsevents_send_event( lspmac_bis[i].  

                    changeEventOn);  

            if( bp->position==0 && bp->changeEventOff != NULL  

                && bp->changeEventOff[0] != 0)  

                lsevents_send_event( lspmac_bis[i].  

                    changeEventOff);  

        }  

    }  

    bp->previous = bp->position;  

    pthread_mutex_unlock( &(bp->mutex));  

}  

  

pthread_mutex_lock( &ncurses_mutex);  

  

// acc11c_1   INPUTS  

// mask  bit  

// 0x01  0      M1000  Air pressure OK  

// 0x02  1      M1001  Air bearing OK  

// 0x04  2      M1002  Cryo switch  

// 0x08  3      M1003  Backlight Down  

// 0x10  4      M1004  Backlight Up  

// 0x20  5      M1005  Cryo back request  

// 0x40  6      M1006  Cryo is back  

  

//  

// acc11c_2   INPUTS  

// mask  bit  

// 0x01  0      M1008  Fluor Doctor back  

// 0x02  1      M1009  Sample Detected  

// 0x04  2      M1020  {SC load request}  

// 0x08  3      M1021  {SC move cryo back request}  

// 0x10  4      M1022  {SC sample magnet control}  

// 0x20  5      M1013  Etel Ready  

// 0x40  6      M1014  Etel On  

// 0x80  7      M1015  Etel Init OK  

  

if( md2_status.acc11c_2 & 0x01)  

    mvwprintw( term_status2, 3, 10, "%*s", -8, "Fluor Out");  

else  

    mvwprintw( term_status2, 3, 10, "%*s", -8, "Fluor In");  

  

if( md2_status.acc11c_5 & 0x08)  

    mvwprintw( term_status2, 4, 1, "%*s", -(LS_DISPLAY_WINDOW_WIDTH  

        -2), "Dryer On");  

else  

    mvwprintw( term_status2, 4, 1, "%*s", -(LS_DISPLAY_WINDOW_WIDTH  

        -2), "Dryer Off");  

  

if( md2_status.acc11c_2 & 0x02)  

    mvwprintw( term_status2, 2, 1, "%*s", -(LS_DISPLAY_WINDOW_WIDTH  

        -2), "Cap Dectected");  

else  

    mvwprintw( term_status2, 2, 1, "%*s", -(LS_DISPLAY_WINDOW_WIDTH  

        -2), "Cap Not Dectected");  

wnoutrefresh( term_status2);  

  

// acc11c_3   INPUTS  

// mask  bit  

// 0x01  0      M1025  Minikappa OK  

// 0x02  1      M1023  {SC unload request}  

// 0x04  2      M1024  Smartmagnet is on (note in pmc saying this is not used  

//                      in VB interface)  

// 0x08  3      M1027  Arm Parked  

// 0x10  4      M1031  Smartmagnet error (coil is broken)

```

```

// 0x20  5
// 0x40  6
// 0x80  7
// 0x100 8    M1048  Shutter is open (note in pmc says: slow input !!!)

// acc11c_4  INPUTS
// mask bit
// 0x01 0   M1031 {laser mirror is back}
// 0x02 1   M1032 {laser PSS OK}
// 0x04 2   M1033 {laser shutter open}

// acc11c_5  OUTPUTS
// mask bit
// 0x01 0   M1100 Mag Off
// 0x02 1   M1191 Condenser Out
// 0x04 2   M1102 Cryo Back
// 0x08 3   M1103 Dryer On
// 0x10 4   M1104 FluoDet Out
// 0x20 5   M1105 {smartmagnet on/off: note in pmc says this is not used}
// 0x40 6   M1106 1=SmartMag, 0=Permanent Mag
// 

if( md2_status.acc11c_5 & 0x04)
    mvwprintw( term_status2, 3, 1, "%*s", -8, "Cryo Out");
else
    mvwprintw( term_status2, 3, 1, "%*s", -8, "Cryo In ");

// acc11c_6  OUTPUTS
// mask bit
// 0x0001 0 M1040 {SC Sample transfer is on}
// 0x0002 1
// 0x0004 2
// 0x0008 3
// 0x0010 4
// 0x0020 5
// 0x0040 6
// 0x0080 7 M1115 Etel Enable
// 0x0100 8 M1124 Fast Shutter Enable
// 0x0200 9 M1125 Fast Shutter Manual Enable
// 0x0400 10 M1126 Fast Shutter On
// 0x0800 11
// 0x1000 12 M1128 ADC1 gain bit 0
// 0x2000 13 M1129 ADC1 gain bit 1
// 0x4000 14 M1130 ADC2 gain bit 0
// 0x8000 15 M1131 ADC2 gain bit 1
// 

if( md2_status.acc11c_5 & 0x02)
    mvwprintw( term_status, 3, 1, "%*s", -(LS_DISPLAY_WINDOW_WIDTH
        -2), "Backlight Up");
else
    mvwprintw( term_status, 3, 1, "%*s", -(LS_DISPLAY_WINDOW_WIDTH
        -2), "Backlight Down");
mvwprintw( term_status, 4, 1, "Front: %*u",
    LS_DISPLAY_WINDOW_WIDTH-2-8, (int)flight->position);
mvwprintw( term_status, 5, 1, "Back: %*u", LS_DISPLAY_WINDOW_WIDTH
    -2-7, (int)blight->position);
mvwprintw( term_status, 6, 1, "Piezo: %*u",
    LS_DISPLAY_WINDOW_WIDTH-2-8, (int)fscint->position);
wnoutrefresh( term_status);

wnoutrefresh( term_input);
doupdate();
pthread_mutex_unlock( &ncurses_mutex);

#ifdef SHOW_RATE
if( ++cnt % 1000 == 0) {
    long diff_sec;
    long diff_nsec;

    clock_gettime( CLOCK_REALTIME, &ts2);

    diff_sec = ts2.tv_sec - ts1.tv_sec;
    diff_nsec = ts2.tv_nsec - ts1.tv_nsec;

    if( diff_nsec < 0) {
        diff_nsec += 1000000000;
        diff_sec--;
    }

    lslogging_log_message( "Refresh Rate: %0.1f Hz", (
        double)cnt / (diff_sec + diff_nsec/1000000000.));

    cnt = 0;
}

```

```

}
#endif
}

```

#### 7.5.4.29 void lspmac\_GetAllIVars( )

Request the values of all the I variables.

Definition at line 2139 of file lspmac.c.

```

{
static char *cmds = "IO..8191";
lspmac_send_command( VR_DOWNLOAD,
    VR_PMAC_SENDLINE, 0, 0, strlen( cmds), cmds,
    lspmac_GetAllIVarsCB, 0, NULL);
}

```

#### 7.5.4.30 void lspmac\_GetAllIVarsCB ( pmac\_cmd\_queue\_t \* cmd, int nreceived, char \* buff )

Receive the values of all the I variables Update our Postgresql database with the results.

##### Parameters

in	<i>cmd</i>	The command that gave this response
in	<i>nreceived</i>	Number of bytes received
in	<i>buff</i>	The byte buffer

Definition at line 2122 of file lspmac.c.

```

{
static char qs[LS_PG_QUERY_STRING_LENGTH];
char *sp;
int i;
for( i=0, sp=strtok(buff, "\r"); sp != NULL; sp=strtok( NULL, "\r"), i++) {
    snprintf( qs, sizeof( qs)-1, "SELECT pmac.md2_ivar_set( %d, '%s')", i, sp);
    qs[sizeof( qs)-1]=0;
    lspg_query_push( NULL, qs);
}
}

```

#### 7.5.4.31 void lspmac\_GetAllMVars( )

Request the values of all the M variables.

Definition at line 2164 of file lspmac.c.

```

{
static char *cmds = "MO..8191->";
lspmac_send_command( VR_DOWNLOAD,
    VR_PMAC_SENDLINE, 0, 0, strlen( cmds), cmds,
    lspmac_GetAllMVarsCB, 0, NULL);
}

```

#### 7.5.4.32 void lspmac\_GetAllMVarsCB ( pmac\_cmd\_queue\_t \* cmd, int nreceived, char \* buff )

Receive the values of all the M variables Update our database with the results.

##### Parameters

in	<i>cmd</i>	The command that started this
in	<i>nreceived</i>	Number of bytes received
in	<i>buff</i>	Our byte buffer

Definition at line 2147 of file lspmac.c.

```

    {
static char qs[LS_PG_QUERY_STRING_LENGTH];
char *sp;
int i;
for( i=0, sp=strtok(buff, "\r"); sp != NULL; sp=strtok( NULL, "\r"), i++) {
    snprintf( qs, sizeof( qs)-1, "SELECT pmac.md2_mvar_set( %d, '%s')", i, sp);
    qs[sizeof( qs)-1]=0;
    lspg_query_push( NULL, qs);
}
}

```

#### 7.5.4.33 int lspmac\_getBIPosition ( *Ispmac.bi\_t* \* *bip* )

get binary input value

Definition at line 1649 of file lspmac.c.

```

    {
int rtn;
pthread_mutex_lock( &bip->mutex );
rtn = bip->position;
pthread_mutex_unlock( &bip->mutex );
return rtn;
}

```

#### 7.5.4.34 void lspmac\_Getmem ( )

Request a block of double buffer memory.

Definition at line 1146 of file lspmac.c.

```

    {
int nbytes;
nbytes = (dbmemIn + 1400 > sizeof( dbmem) ) ? sizeof( dbmem)
    - dbmemIn : 1400;
lspmac_SockGetmem( dbmemIn, nbytes);
}

```

#### 7.5.4.35 void lspmac\_GetmemReplyCB ( *pmac\_cmd\_queue\_t* \* *cmd*, int *nreceived*, char \* *buff* )

Service a reply to the getmem command.

##### Parameters

<i>cmd</i>	Queue item this is a reply to
<i>nreceived</i>	Number of bytes received
<i>buff</i>	Buffer of bytes received

Definition at line 1070 of file lspmac.c.

```

    {
memcpy( &(dbmem[ntohs(cmd->pcmd.wValue)]), buff, nreceived);

dbmemIn += nreceived;
if( dbmemIn >= sizeof( dbmem) ) {
    dbmemIn = 0;
}
}

```

7.5.4.36 double lspmac\_getPosition ( *Lspmac\_motor\_t* \* *mp* )

get the motor position (with locking)

## Parameters

<i>mp</i>	the motor object
-----------	------------------

Definition at line 1402 of file lspmac.c.

```
{
double rtn;
pthread_mutex_lock( &(mp->mutex));
rtn = mp->position;
pthread_mutex_unlock( &(mp->mutex));
return rtn;
}
```

7.5.4.37 void lspmac\_GetShortReplyCB ( *pmac\_cmd\_queue\_t* \* *cmd*, int *nreceived*, char \* *buff* )

Receive a reply that does not require multiple buffers.

## Parameters

in	<i>cmd</i>	Queue item this is a reply to
in	<i>nreceived</i>	Number of bytes received
in	<i>buff</i>	The buffer of bytes

Definition at line 1010 of file lspmac.c.

```
{
char *sp;      // pointer to the command this is a reply to
char *tmp;

if( nreceived < 1400)
    buff[nreceived]=0;

sp = (char *) (cmd->pcmd.bData);

if( *buff == 0) {
    lslogging_log_message( "%s", sp);
} else {
    tmp = cleanstr( buff);
    lslogging_log_message( "%s: %s", sp, tmp);
    free( tmp);
}

memset( cmd->pcmd.bData, 0, sizeof( cmd->pcmd.bData));
}
```

7.5.4.38 void lspmac\_home1\_queue ( *Lspmac\_motor\_t* \* *mp* )

Home the motor.

## Parameters

in	<i>mp</i>	motor we are concerned about
----	-----------	------------------------------

Definition at line 1272 of file lspmac.c.

```
{
int i;
int motor_num;
int coord_num;
```

```

char **home;

pthread_mutex_lock( &(mp->mutex));

motor_num = lsredis_get1( mp->motor_num);
coord_num = lsredis_get1( mp->coord_num);
home      = lsredis_get_string_array( mp->home);

// Each of the motors should have this defined
// but let's not seg fault if home is missing
//
if( home == NULL || *home == NULL) {
//
// Note we are already initialized
// so if we are here there is something wrong.
//
lslogging_log_message( "lspmac_home1_queue: null or
    empty home strings for motor %s", mp->name);
pthread_mutex_unlock( &(mp->mutex));
return;
}

// We've already been called.  Don't home again until
// we're finish with the last time.
//
if( mp->homimg) {
    pthread_mutex_unlock( &(mp->mutex));
    return;
}

//
// Don't go on if any other motors in this coordinate system are homing.
// It's possible to write the homing program to home all the motors in the
// coordinate
// system.  TODO (hint hint)
//
if( coord_num > 0) {
for( i=0; i<lspmac_nmotors; i++) {
    if( &(lspmac_motors[i]) == mp)
        continue;
    if( lsredis_get1(lspmac_motors[i].coord_num) ==
coord_num) {
        int nogo;
        nogo = 0;
        pthread_mutex_lock( &(lspmac_motors[i].mutex));
        //
        // Don't go on if
        //
        // we are homing      or      ( not in position
        while      in open loop)
        //
        if( lspmac_motors[i].homimg || (((lspmac_motors
[i].status2 & 0x01)==0) && ((lspmac_motors[i].status1 & 0x040000)
!= 0)))
            nogo = 1;
        pthread_mutex_unlock( &(lspmac_motors[i].mutex));
        if( nogo) {
            pthread_mutex_unlock( &(mp->mutex));
            return;
        }
    }
}
mp->homimg   = 1;
mp->not_done = 1;      // set up waiting for cond
mp->motion_seen = 0;
// This opens the control loop.
// The status routine should notice this and the fact that
// the homing flag is set and call on the home2 routine
//
// Only send the open loop command if we are not in
// open loop mode already.  This test might prevent a race condition
// where we've already moved the home2 routine (and queue the homing program
// motion)
// before the open loop command is dequeued and acted on.
//
if( ~(mp->status1) & 0x040000) {
    lspmac_SockSendDPLine( mp->name, "#%d$*",
    motor_num);
}

pthread_mutex_unlock( &(mp->mutex));
lsevents_send_event( "%s Homing", mp->name);
}

```

7.5.4.39 void lspmac\_home2\_queue( *Lspmac\_motor\_t* \* *mp* )

Second stage of homing.

## Parameters

in	<i>mp</i>	motor we are concerned about
----	-----------	------------------------------

Definition at line 1360 of file lspmac.c.

```
{
char **spp;
char **home;

// At this point we are in open loop.
// Run the motor specific commands
//

pthread_mutex_lock( &(mp->mutex) );

home = lsredis_get_string_array( mp->home );

// We don't have any motors that have a null home text array so
// there is currently no need to worry about this case other than
// not to seg fault
//
// Also, Only go on if the first homing phase has been started
//
if( home == NULL || mp->homming != 1) {
    pthread_mutex_unlock( &(mp->mutex));
    return;
}

for( spp = home; *spp != NULL; spp++) {
    lslogging_log_message( "home2 is queuing '%s'\n", *spp
    );
    lspmac_SockSendDPLine( mp->name, *spp);
}

mp->homming = 2;
pthread_mutex_unlock( &(mp->mutex));
}
```

7.5.4.40 void lspmac\_init( int *ivarsflag*, int *mvarsflag* )

Initialize this module.

## Parameters

in	<i>ivarsflag</i>	Set global flag to harvest i variables
in	<i>mvarsflag</i>	Set global flag to harvest m variables

Definition at line 3799 of file lspmac.c.

```
{
static int first_time = 1;
int i;
int err;
ENTRY entry_in, *entry_outp;
md2_status_t *p;
pthread_mutexattr_t mutex_initializer;

if( first_time) {
    // Set our global harvest flags
    getivars = ivarsflag;
    getmvars = mvarsflag;

    // Use recursive mutexes
    //
```

```

pthread_mutexattr_init( &mutex_initializer);
pthread_mutexattr_settype( &mutex_initializer, PTHREAD_MUTEX_RECURSIVE);

// All important status mutex
pthread_mutex_init( &md2_status_mutex, &mutex_initializer);

//
// Get the MD2 initialization strings
//
// lspmac_md2_init = lsredis_get_obj( "md2_pmac.init"); // hard coded
now.

//
// Initialize the motor objects
//

p = &md2_status;

omega = lspmac_motor_init( &(lspmac_motors
[ 0]), 0, 0, &p->omega_act_pos, &p->omega_status_1
, &p->omega_status_2, "Omega #1 &1 X", "omega",
lspmac_moveabs_queue, lspmac_jogabs_queue
);
alignx = lspmac_motor_init( &(lspmac_motors
[ 1]), 0, 1, &p->alignx_act_pos, &p->alignx_status_1
, &p->alignx_status_2, "Align X #2 &3 X", "align.x",
lspmac_moveabs_queue, lspmac_jogabs_queue
);
aligny = lspmac_motor_init( &(lspmac_motors
[ 2]), 0, 2, &p->aligny_act_pos, &p->aligny_status_1
, &p->aligny_status_2, "Align Y #3 &3 Y", "align.y",
lspmac_moveabs_queue, lspmac_jogabs_queue
);
alignz = lspmac_motor_init( &(lspmac_motors
[ 3]), 0, 3, &p->alignz_act_pos, &p->alignz_status_1
, &p->alignz_status_2, "Align Z #4 &3 Z", "align.z",
lspmac_moveabs_queue, lspmac_jogabs_queue
);
anal = lspmac_motor_init( &(lspmac_motors
[ 4]), 0, 4, &p->analyzer_act_pos, &p->analyzer_status_1
, &p->analyzer_status_2, "Anal #5", "lightPolar",
lspmac_moveabs_queue, lspmac_jogabs_queue
);
zoom = lspmac_motor_init( &(lspmac_motors
[ 5]), 1, 0, &p->zoom_act_pos, &p->zoom_status_1
, &p->zoom_status_2, "Zoom #6 &4 Z", "cam.zoom",
lspmac_movezoom_queue, lspmac_movezoom_queue
);
apery = lspmac_motor_init( &(lspmac_motors
[ 6]), 1, 1, &p->aperturey_act_pos, &p->aperturey_status_1
, &p->aperturey_status_2, "Aper Y #7 &5 Y", "appy",
lspmac_moveabs_queue, lspmac_jogabs_queue
);
aperz = lspmac_motor_init( &(lspmac_motors
[ 7]), 1, 2, &p->aperturez_act_pos, &p->aperturez_status_1
, &p->aperturez_status_2, "Aper Z #8 &5 Z", "appz",
lspmac_moveabs_queue, lspmac_jogabs_queue
);
capy = lspmac_motor_init( &(lspmac_motors
[ 8]), 1, 3, &p->capy_act_pos, &p->capy_status_1
, &p->capy_status_2, "Cap Y #9 &5 U", "capy",
lspmac_moveabs_queue, lspmac_jogabs_queue
);
capz = lspmac_motor_init( &(lspmac_motors
[ 9]), 1, 4, &p->capz_act_pos, &p->capz_status_1
, &p->capz_status_2, "Cap Z #10 &5 V", "capz",
lspmac_moveabs_queue, lspmac_jogabs_queue
);
scint = lspmac_motor_init( &(lspmac_motors
[10]), 2, 0, &p->scint_act_pos, &p->scint_status_1
, &p->scint_status_2, "Scin Z #11 &5 W", "scint",
lspmac_moveabs_queue, lspmac_jogabs_queue
);
cenx = lspmac_motor_init( &(lspmac_motors
[11]), 2, 1, &p->centerx_act_pos, &p->centerx_status_1
, &p->centerx_status_2, "Cen X #17 &2 X", "centering.x",
lspmac_moveabs_queue, lspmac_jogabs_queue
);
ceny = lspmac_motor_init( &(lspmac_motors
[12]), 2, 2, &p->centery_act_pos, &p->centery_status_1
, &p->centery_status_2, "Cen Y #18 &2 Y", "centering.y",
lspmac_moveabs_queue, lspmac_jogabs_queue
);
kappa = lspmac_motor_init( &(lspmac_motors
[13]), 2, 3, &p->kappa_act_pos, &p->kappa_status_1
, &p->kappa_status_2, "Kappa #19 &7 X", "kappa",
lspmac_moveabs_queue, lspmac_jogabs_queue

```

```

);
phi    = lspmacc_motor_init( &(lspmacc_motors
[14]), 2, 4, &p->phi_act_pos,           &p->phi_status_1,
&p->phi_status_2,      "Phi #20 &7 Y", "phi",
lspmacc_moveabs_queue, lspmacc_jogabs_queue
);

fshut  = lspmacc_fshut_init( &(lspmacc_motors
[15]));
flight = lspmacc_dac_init( &(lspmacc_motors
[16]), &p->front_dac,      "frontLight.intensity",
lspmacc_movedac_queue);
blight = lspmacc_dac_init( &(lspmacc_motors
[17]), &p->back_dac,       "backLight.intensity",
lspmacc_movedac_queue);
fscint = lspmacc_dac_init( &(lspmacc_motors
[18]), &p->scint_piezo, "M1203", "scint.focus",
lspmacc_movedac_queue);

smart_mag_oo = lspmacc_bo_init( &(lspmacc_motors
[19]), "smartMagnet", "M1100=%d", &(md2_status.accllc_5), 0x01)
;
blight_ud   = lspmacc_bo_init( &(lspmacc_motors
[20]), "backLight",      "M1101=%d", &(md2_status.accllc_5), 0x02)
;
cryo        = lspmacc_bo_init( &(lspmacc_motors
[21]), "cryo",          "M1102=%d", &(md2_status.accllc_5), 0x04)
;
dryer       = lspmacc_bo_init( &(lspmacc_motors
[22]), "dryer",          "M1103=%d", &(md2_status.accllc_5), 0x08)
;
fluo        = lspmacc_bo_init( &(lspmacc_motors
[23]), "fluo",          "M1104=%d", &(md2_status.accllc_5), 0x10)
;
flight_oo   = lspmacc_soft_motor_init( &(lspmacc_motors[24]),
"frontLight",
lspmacc_moveabs_frontlight_oo_queue);
blight_f    = lspmacc_soft_motor_init( &(lspmacc_motors[25]),
"backLight.factor",
lspmacc_moveabs_bright_factor_queue);
flight_f    = lspmacc_soft_motor_init( &(lspmacc_motors[26]),
"frontLight.factor",
lspmacc_moveabs_flight_factor_queue);

lp_air      = lspmacc_bi_init( &(lspmacc_bis
[ 0]), &(md2_status.accllc_1), 0x01, "Low Pressure Air OK",
"Low Pressure Air Failed");
hp_air      = lspmacc_bi_init( &(lspmacc_bis
[ 1]), &(md2_status.accllc_1), 0x02, "High Pressure Air OK",
"High Pressure Air Failed");
cryo_switch = lspmacc_bi_init( &(lspmacc_bis
[ 2]), &(md2_status.accllc_1), 0x04, "CryoSwitchChanged",
"CryoSwitchChanged");
blight_down = lspmacc_bi_init( &(lspmacc_bis
[ 3]), &(md2_status.accllc_1), 0x08, "Backlight Down",
"Backlight Not Down");
blight_up   = lspmacc_bi_init( &(lspmacc_bis
[ 4]), &(md2_status.accllc_1), 0x10, "Backlight Up",
"Backlight Not Up");
cryo_back   = lspmacc_bi_init( &(lspmacc_bis
[ 5]), &(md2_status.accllc_1), 0x40, "Cryo Back",
"Cryo Not Back");
fluor_back  = lspmacc_bi_init( &(lspmacc_bis
[ 6]), &(md2_status.accllc_2), 0x01, "Fluor. Det. Parked",
"Fluor. Det. Not Parked");
sample_detected = lspmacc_bi_init( &(lspmacc_bis
[ 7]), &(md2_status.accllc_2), 0x02, "SamplePresent",
"SampleAbsent");
etel_ready   = lspmacc_bi_init( &(lspmacc_bis
[ 8]), &(md2_status.accllc_2), 0x20, "ETEL Ready",
"ETEL Not Ready");
etel_on      = lspmacc_bi_init( &(lspmacc_bis
[ 9]), &(md2_status.accllc_2), 0x40, "ETEL On",
"ETEL Off");
etel_init_ok = lspmacc_bi_init( &(lspmacc_bis
[10]), &(md2_status.accllc_2), 0x80, "ETEL Init OK",
"ETEL Init Not OK");
minikappa_ok = lspmacc_bi_init( &(lspmacc_bis
[11]), &(md2_status.accllc_3), 0x01, "Minikappa OK",
"Minikappa Not OK");
smart_mag_on = lspmacc_bi_init( &(lspmacc_bis
[12]), &(md2_status.accllc_3), 0x04, "Smart Magnet On",
"Smart Magnet Not On");
arm_parked   = lspmacc_bi_init( &(lspmacc_bis
[13]), &(md2_status.accllc_3), 0x08, "Arm Parked",
"Arm Not Parked");
smart_mag_err = lspmacc_bi_init( &(lspmacc_bis

```

```

[14]), &(md2_status.accllc_3), 0x10, "Smart Magnet Error",
"Smart Magnet OK");
shutter_open    = lspmac_bi_init( &lspmac_bis
[15]), &(md2_status.accllc_3), 0x100, "Shutter Open",
"Shutter Not Open");
smart_mag_off   = lspmac_bi_init( &lspmac_bis
[16]), &(md2_status.accllc_5), 0x01, "Smart Magnet Off",
"Smart Magnet Not Off");

// Set up hash table
//
err = hcreate_r( LSPMAC_MAX_MOTORS * 2, &motors_ht
);
if( err == 0) {
lslogging_log_message( "lspmac_init: hcreate_r
failed: '%s'", strerror( errno));
exit( -1);
}
for( i=0; i<lspmac_nmotors; i++) {
entry_in.key    = lspmac_motors[i].name;
entry_in.data   = &lspmac_motors[i];
err = hsearch_r( entry_in, ENTER, &entry_outp, &motors_ht);
if( err == 0) {
lslogging_log_message( "lspmac_init: hsearch_r
failed for motor %s: '%s'", lspmac_motors[i].name, strerror( errno));
exit( -1);
}
}

//
// Initialize several commands that get called, perhaps, alot
//
rr_cmd.RequestType = VR_UPLOAD;
rr_cmd.Request      = VR_PMAC_READREADY;
rr_cmd.wValue       = 0;
rr_cmd.wIndex       = 0;
rr_cmd.wLength      = htons(2);
memset( rr_cmd.bData, 0, sizeof(rr_cmd.bData));

gb_cmd.RequestType = VR_UPLOAD;
gb_cmd.Request      = VR_PMAC_GETBUFFER;
gb_cmd.wValue       = 0;
gb_cmd.wIndex       = 0;
gb_cmd.wLength      = htons(1400);
memset( gb_cmd.bData, 0, sizeof(gb_cmd.bData));

cr_cmd.RequestType = VR_UPLOAD;
cr_cmd.Request      = VR_CTRL_RESPONSE;
cr_cmd.wValue       = 0;
cr_cmd.wIndex       = 0;
cr_cmd.wLength      = htons(1400);
memset( cr_cmd.bData, 0, sizeof(cr_cmd.bData));

//
// Initialize some mutexes and conditions
//
pthread_mutex_init( &pmac_queue_mutex, &mutex_initializer);
pthread_cond_init( &pmac_queue_cond, NULL);

lspmac_shutter_state = 0;
// assume the shutter is now closed: not a big deal if we are wrong
pthread_mutex_init( &lspmac_shutter_mutex, &
mutex_initializer);
pthread_cond_init( &lspmac_shutter_cond, NULL);
pmacfd.fd = -1;

pthread_mutex_init( &lspmac_moving_mutex, &
mutex_initializer);
pthread_cond_init( &lspmac_moving_cond, NULL);

pthread_mutex_init( &lspmac_ascii_mutex, &
mutex_initializer);

pthread_mutex_init( &lspmac_ascii_buffers_mutex,
&mutex_initializer);

lsevents_preregister_event( "omega crossed zero")
;
lsevents_preregister_event( "Move Aborted");
lsevents_preregister_event( "Combined Move
Aborted");
lsevents_preregister_event( "Abort Request queued"

```

```

        ");
lsevents_preregister_event( "Abort Request
accepted");
lsevents_preregister_event( "Reset queued");
lsevents_preregister_event( "Reset command
accepted");

for( i=1; i<=16; i++) {
lsevents_preregister_event( "Coordsys %d
Stopped", i);
}
first_time = 0;
}

// clear the ascii communications buffers
//
{
    uint32_t cc;
    cc = 0;
lspmac_send_command( VR_UPLOAD, VR_PMAC_SETMEM
, 0x0e9e, 0, 4, (char *)&cc, NULL, 1, NULL);

    cc = 0x18;
lspmac_send_command( VR_UPLOAD, VR_PMAC_SETMEM
, 0x0e9e, 0, 4, (char *)&cc, NULL, 1, NULL);
}

lspmac_SockSendDPLine( NULL, "I5=0");
lspmac_SockSendDPLine( NULL, "ENABLE PLCC 0,2");
lspmac_SockSendDPLine( NULL, "DISABLE PLCC 1");
lspmac_SockSendDPLine( NULL, "I5=3");
}

}

```

#### 7.5.4.41 int lspmac\_jogabs\_queue ( **lspmac\_motor\_t** \* *mp*, double *requested\_position* )

Use jog to move motor to requested position.

##### Parameters

in	<i>mp</i>	The motor to move
in	<i>requested_position</i>	Where to move it

Definition at line 3483 of file lspmac.c.

```

{

return lspmac_move_or_jog_abs_queue( mp,
requested_position, 1);
}
```

#### 7.5.4.42 void lspmac\_light\_zoom\_cb ( char \* *event* )

Set the backlight intensity whenever the zoom is changed (and the backlight is up)

##### Parameters

<i>event</i>	Name of the event that called us
--------------	----------------------------------

Definition at line 4065 of file lspmac.c.

```

int z;

pthread_mutex_lock( &zoom->mutex );
z = zoom->requested_position;
pthread_mutex_unlock( &zoom->mutex );
```

```

lslogging_log_message( "lspmac_light_zoom_cb: zoom = %d"
, z);

if( lspmac_getPosition( flight_o ) != 0.0) {
    flight->moveAbs( flight, (double)z);
} else {
    flight->moveAbs( flight, 0.0);
}
if( lspmac_getPosition( blight_ud ) != 0.0) {
    blight->moveAbs( blight, (double)z);
} else {
    blight->moveAbs( blight, 0.0);
}
}

```

#### 7.5.4.43 double lspmac\_lut( int nlut, double \* lut, double x )

Look up table support for motor positions (think x=zoom, y=light intensity) use a lookup table to find the "counts" to move the motor to the requested position. The look up table is a simple one dimensional array with the x values as even indicies and the y values as odd indicies.

Returns: y value

##### Parameters

in	<i>nlut</i>	number of entries in lookup table
in	<i>lut</i>	The lookup table: even indicies are the x values, odd are the y's
in	<i>x</i>	The x value we are looking up.

Definition at line 397 of file lspmac.c.

```

{
int i, foundone;
double m;
double y1, y2, x1, x2, y;

foundone = 0;
if( lut != NULL && nlut > 1) {
    for( i=0; i < 2*nlut; i += 2) {
        x1 = lut[i];
        y1 = lut[i+1];
        if( i < 2*nlut - 2) {
            x2 = lut[i+2];
            y2 = lut[i+3];
        }
        //
        // First one too big? Use the y value of the first element
        //
        if( i == 0 && x1 > x) {
            y = y1;
            foundone = 1;
            break;
        }
        //
        // Look for equality
        //
        if( x1 == x) {
            y = y1;
            foundone = 1;
            break;
        }
        //
        // Maybe interpolate
        //
        if( (i < 2*nlut-2) && x < x2) {
            m = (y2 - y1) / (x2 - x1);
            y = m*(x - x1) + y1;
            foundone = 1;
            break;
        }
    }
    if( foundone == 0) {
        // must be bigger than the last entry
    }
}

```

```

    //
    //
    y = lut[2*(nlut-1) + 1];
}
return y;
}
return 0.0;
}

```

#### 7.5.4.44 void lspmacc\_more\_ascii\_cb ( pmac\_cmd\_queue\_t \* cmd, int nreceived, char \* buff )

we are expecting more characters from the DPRAM ASCII interface

Definition at line 1890 of file lspmacc.c.

```

lspmacc_get_ascii( cmd->event );
{

```

#### 7.5.4.45 lspmacc\_motor\_t\* lspmacc\_motor\_init ( lspmacc\_motor\_t \* d, int wy, int wx, int \* posp, int \* stat1p, int \* stat2p, char \* wtitle, char \* name, int(\*) (lspmacc\_motor\_t \*, double) moveAbs, int(\*) (lspmacc\_motor\_t \*, double) jogAbs )

Initialize a pmac stepper or servo motor.

##### Parameters

in,out	<i>d</i>	An uninitialized motor object
in	<i>wy</i>	Curses status window row index
in	<i>wx</i>	Curses status window column index
in	<i>posp</i>	Pointer to position status
in	<i>stat1p</i>	Pointer to 1st status word
in	<i>stat2p</i>	Pointer to 2nd status word
in	<i>wtitle</i>	Title for this motor (to display)
in	<i>name</i>	This motor's name
in	<i>moveAbs</i>	Method to use to move this motor (motion program preferred)
in	<i>jogAbs</i>	Method to use to jog this motor (jog preferred)

Definition at line 3631 of file lspmacc.c.

```

{
    _lspmacc_motor_init( d, name );

    d->moveAbs      = moveAbs;
    d->jogAbs       = jogAbs;
    d->read          = lspmacc_pmacmotor_read;
    d->actual_pos_ctns_p = posp;
    d->status1_p     = stat1p;
    d->status2_p     = stat2p;

    d->win = newwin( LS_DISPLAY_WINDOW_HEIGHT,
                      LS_DISPLAY_WINDOW_WIDTH, wy*LS_DISPLAY_WINDOW_HEIGHT
                      , wx*LS_DISPLAY_WINDOW_WIDTH);

    pthread_mutex_lock( &ncurses_mutex );
    box( d->win, 0, 0 );
    mvwprintw( d->win, 1, 1, "%s", wtitle );
    wnoutrefresh( d->win );
    pthread_mutex_unlock( &ncurses_mutex );

    lsevents_preregister_event( "%s Homing",           d->
                                name );
    lsevents_preregister_event( "%s Homed",            d->
                                name );
    lsevents_preregister_event( "%s Moving",           d->
                                name );
    lsevents_preregister_event( "%s In Position",      d->
                                name );
}
```

```

    name);
lsevents_preregister_event( "%s Move Aborted", d->
    name);

    return d;
}

```

#### 7.5.4.46 int lspmact\_move\_or\_jog\_abs\_queue ( lspmact\_motor\_t \* mp, double requested\_position, int use\_jog )

Move method for normal stepper and servo motor objects Returns non-zero on abort, zero if OK.

< format string for coordinate system move

< coordinate system bit

< the requested position in units of "counts"

< motor and coordinate system;

< our axis

#### Parameters

in	<i>mp</i>	The motor to move
in	<i>requested_-position</i>	Where to move it
in	<i>use_jog</i>	1 to force jog, 0 for motion prog

Definition at line 3235 of file lspmact.c.

```

{
char *fmt;
int q100;
int requested_pos_cnts;
int coord_num, motor_num;
char *axis;
double u2c;
double neutral_pos;
double min_pos, max_pos;
int pos_limit_hit, neg_limit_hit, in_position_band;
struct timespec timeout, now;
int err;

pthread_mutex_lock( &(mp->mutex));

u2c      = lsredis_getd( mp->u2c);
motor_num = lsredis_getl( mp->motor_num);
coord_num = lsredis_getl( mp->coord_num);
axis     = lsredis_getstr( mp->axis);
neutral_pos = lsredis_getd( mp->neutral_pos);
min_pos   = lsredis_getd( mp->min_pos) -
            neutral_pos;
max_pos   = lsredis_getd( mp->max_pos) -
            neutral_pos;
pos_limit_hit = lsredis_getd( mp->pos_limit_hit
    );
neg_limit_hit = lsredis_getd( mp->neg_limit_hit
    );
in_position_band = lsredis_getl( mp->in_position_band
    );

if( u2c == 0.0 || requested_position < min_pos || requested_position >
    max_pos) {
    //
    // Shouldn't try moving a motor that's in trouble
    //
    pthread_mutex_unlock( &(mp->mutex));
    lslogging_log_message( "lspmact_move_or_jog_abs_queue:
        %s u2c=%f requested position=%f min allowed=%f max allowed=%f", mp->name
        , u2c, requested_position, min_pos, max_pos);
    lsevents_send_event( "%s Move Aborted", mp->name);
    return 1;
}

if( (neg_limit_hit && (requested_position < mp->position)) || (pos_limit_hit
    && (requested_position > mp->position))) {

```

```

pthread_mutex_unlock( &(mp->mutex));
lslogging_log_message( "lspmacc_move_or_jog_abs_queue:
    %s Moving wrong way on limit: requested position=%f  current position=%f  low
    limit=%d high limit=%d",
    mp->name, requested_position, mp->position
    , neg_limit_hit, pos_limit_hit);
lsevents_send_event( "%s Move Aborted", mp->name);
return 2;
}

mp->requested_position = requested_position;
if( mp->nlut > 0 && mp->lut != NULL) {
    mp->requested_pos_cnts = lspmacc_lut( mp->nlut
        , mp->lut, requested_position);
} else {
    mp->requested_pos_cnts = u2c * (requested_position +
        neutral_pos);
}
requested_pos_cnts = mp->requested_pos_cnts;

//
// Bluff if we are already there
//
if( (abs( requested_pos_cnts - mp->actual_pos_cnts) * 16 <
    in_position_band) || (lsredis_getb( mp->active) != 1)) {
    //
    // Lie and say we moved even though we didn't. Who will know? We are
    // within the deadband or not active.
    //
    mp->not_done      = 1;
    mp->motion_seen   = 0;
    mp->command_sent  = 0;

    lsevents_send_event( "%s Moving", mp->name);

    mp->not_done      = 0;
    mp->motion_seen   = 1;
    mp->command_sent  = 1;

    if( lsredis_getb( mp->active) != 1) {
        //
        // fake the motion for simulated motors
        //
        mp->position = requested_position;
        mp->actual_pos_cnts = requested_pos_cnts;
    }
    pthread_mutex_unlock( &(mp->mutex));

    lsevents_send_event( "%s In Position", mp->name);
    return 0;
}

mp->not_done      = 1;
mp->motion_seen   = 0;
mp->command_sent  = 0;

if( use_jog || axis == NULL || *axis == 0) {
    use_jog = 1;
} else {
    use_jog = 0;
    q100 = 1 << (coord_num -1);
}

pthread_mutex_unlock( &(mp->mutex));

if( !use_jog) {
    //
    // Make sure the coordinate system is not moving something, wait if it is
    //
    pthread_mutex_lock( &lspmacc_moving_mutex);

    clock_gettime( CLOCK_REALTIME, &now);
    //
    // TODO: Have all moves estimate how long they'll take and use that here
    //
    timeout.tv_sec  = now.tv_sec + 60.0;           // a long timeout, but
    // we might really be moving something that takes this long (or longer)
    timeout.tv_nsec = now.tv_nsec;

    err = 0;
    while( err == 0 && (lspmacc_moving_flags & q100) != 0)
        err = pthread_cond_timedwait( &lspmacc_moving_cond, &

```

```

lspmac_moving_mutex, &timeout);

pthread_mutex_unlock( &lspmac_moving_mutex);

if( err == ETIMEDOUT) {
    lslogging_log_message( "
    lspmac_move_or_jog_abs_queue: Timed Out.  lspmac_moving_flags = %0x", lspmac_moving_flags
);
    lsevents_send_event( "%s Move Aborted", mp->name);
    return 1;
}

//
// Set the "we are moving this coordinate system" flag
//
lspmac_SockSendDPLine( NULL, "M5075=(M5075 | %d)",
q100);

switch( *axis) {
case 'A':
    fmt = "%d Q16=%d Q100=%d B146R";
    break;

case 'B':
    fmt = "%d Q17=%d Q100=%d B147R";
    break;

case 'C':
    fmt = "%d Q18=%d Q100=%d B148R";
    break;
case 'X':
    fmt = "%d Q10=%d Q100=%d B140R";
    break;

case 'Y':
    fmt = "%d Q11=%d Q100=%d B141R";
    break;

case 'Z':
    fmt = "%d Q12=%d Q100=%d B142R";
    break;

case 'U':
    fmt = "%d Q13=%d Q100=%d B143R";
    break;

case 'V':
    fmt = "%d Q14=%d Q100=%d B144R";
    break;

case 'W':
    fmt = "%d Q15=%d Q100=%d B145R";
    break;
}

//
// Make sure the flag has been seen
//

clock_gettime( CLOCK_REALTIME, &now);
timeout.tv_sec = now.tv_sec + 4.0;           // also a long timeout.
    This should really only take a few milliseconds on a slow day
timeout.tv_nsec = now.tv_nsec;

pthread_mutex_lock( &lspmac_moving_mutex);

err = 0;
while( err == 0 && (lspmac_moving_flags & q100) == 0)
    err = pthread_cond_timedwait( &lspmac_moving_cond, &
        lspmac_moving_mutex, &timeout);
pthread_mutex_unlock( &lspmac_moving_mutex);

if( err == ETIMEDOUT) {
    lslogging_log_message( "
    lspmac_move_or_jog_abs_queue: Did not see flag propagate.  Move aborted.");
    lsevents_send_event( "%s Move Aborted", mp->name);
    return 1;
}

pthread_mutex_lock( &(mp->mutex));
if( use_jog) {
    lspmac_SockSendDPLine( mp->name, "#%d j=%d",
        motor_num, requested_pos_cnts);
} else {
    lspmac_SockSendDPLine( mp->name, fmt, coord_num,
        requested_pos_cnts, q100);
}

```

```

}
pthread_mutex_unlock( &(mp->mutex));
free( axis);
return 0;
}

```

#### 7.5.4.47 int lspmacc\_move\_or\_jog\_preset\_queue( lspmacc\_motor\_t \* mp, char \* preset, int use\_jog )

move using a preset value returns 0 on success, non-zero on error

##### Parameters

in	<i>mp</i>	Our motor
in	<i>preset</i>	the name of the preset
	<i>use_jog</i>	[in] 1 to force jog, 0 to try motion prog

Definition at line 3444 of file lspmacc.c.

```

{
double pos;
int err;
int rtn;

if( preset == NULL || *preset == 0) {
    lsevents_send_event( "%s Move Aborted", mp->name);
    return 0;
}

err = lsredis_find_preset( mp->name, preset, &pos);

if( err != 0)
    rtn = lspmacc_move_or_jog_abs_queue( mp, pos,
        use_jog);
else {
    lsevents_send_event( "%s Move Aborted", mp->name);
    rtn = 1;
}
return rtn;
}
```

#### 7.5.4.48 int lspmacc\_move\_preset\_queue( lspmacc\_motor\_t \* mp, char \* preset\_name )

Move a given motor to one of its preset positions.

No movement if the preset is not found.

##### Parameters

<i>mp</i>	lspmacc motor pointer
<i>preset_name</i>	Name of the preset to use

Definition at line 2473 of file lspmacc.c.

```

{
double pos;
int err;

lslogging_log_message( "lspmacc_move_preset_queue: Called
    with motor %s and preset named '%s'", mp->name, preset_name);

err = lsredis_find_preset( mp->name, preset_name, &pos
    );
if( err == 0)
    return 1;

err = mp->jogAbs( mp, pos);
if( !err)
```

```

lslogging_log_message( "lspmac_move_preset_queue:
    moving %s to preset '%s' (%f)", mp->name, preset_name, pos);
// the abort event should have been sent in moveAbs
//
return err;
}

```

#### 7.5.4.49 int lspmac\_moveabs\_blight\_factor\_queue ( **lspmac\_motor\_t** \* *mp*, double *pos* )

Definition at line 2681 of file lspmac.c.

```

{
char *fmt;

if( pos >= 60 && pos <= 140) {
    pthread_mutex_lock( &(mp->mutex));
    *mp->actual_pos_cnts_p = pos;
    mp->position =
        pos;
    pthread_mutex_unlock( &(mp->mutex));

    pthread_mutex_lock( &(blight->mutex));
    fmt = lsredis_getstr( blight->redis_fmt);
    lsredis_setstr( blight->u2c, fmt, pos / 100.0);
    free( fmt);
    pthread_mutex_unlock( &(blight->mutex));

    blight->moveAbs( blight, lspmac_getPosition
        ( zoom));
}

return 0;
}

```

#### 7.5.4.50 int lspmac\_moveabs\_bo\_queue ( **lspmac\_motor\_t** \* *mp*, double *requested\_position* )

Move method for binary i/o motor objects.

##### Parameters

in	<i>mp</i>	A binary i/o motor object
in	<i>requested_position</i>	a 1 or a 0 request to move

Definition at line 2547 of file lspmac.c.

```

{
pthread_mutex_lock( &(mp->mutex));
mp->requested_position = requested_position == 0.0 ? 0.0 :
    1.0;
mp->requested_pos_cnts = requested_position == 0.0 ? 0 : 1;

if( mp->requested_position == mp->position) {
    //
    // No real move requested
    //
    mp->not_done = 0;
    mp->motion_seen = 1;
    mp->command_sent = 1;
    lsevents_send_event( "%s Moving", mp->name);
    lsevents_send_event( "%s In Position", mp->name);

} else {
    //
    // Go ahead and send the request
    //
    mp->not_done = 1;
    mp->motion_seen = 0;
    mp->command_sent = 0;
    lspmac_SockSendDpline( mp->name, mp->write_fmt
        , mp->requested_pos_cnts);
}

```

```

}

pthread_mutex_unlock( &(mp->mutex));
return 0;
}

```

#### 7.5.4.51 int lspmac\_moveabs\_flight\_factor\_queue ( *Lspmac\_motor\_t* \* *mp*, double *pos* )

Definition at line 2658 of file lspmac.c.

```

{
char *fmt;

if( pos >= 60 && pos <= 140) {
    pthread_mutex_lock( &(mp->mutex));
    *mp->actual_pos_ctns_p = pos;
    mp->position =
        pos;
    pthread_mutex_unlock( &(mp->mutex));

    pthread_mutex_lock( &(flight->mutex));

    fmt = lsredis_getstr( flight->redis_fmt);
    lsredis_setstr( flight->u2c, fmt, pos / 100.0);
    free( fmt);

    pthread_mutex_unlock( &(flight->mutex));

    flight->moveAbs( flight, lspmac_getPosition
        (zoom));
    return 0;
}
return 1;
}

```

#### 7.5.4.52 int lspmac\_moveabs\_frontlight\_oo\_queue ( *Lspmac\_motor\_t* \* *mp*, double *pos* )

"move" frontlight on/off

Definition at line 2645 of file lspmac.c.

```

{
pthread_mutex_lock( &(mp->mutex));
*mp->actual_pos_ctns_p = pos;
mp->position =
    pos;
pthread_mutex_unlock( &(mp->mutex));
if( pos == 0.0) {
    flight->moveAbs( flight, 0.0);
} else {
    flight->moveAbs( flight, lspmac_getPosition
        (zoom));
}
return 0;
}

```

#### 7.5.4.53 int lspmac\_moveabs\_fshut\_queue ( *Lspmac\_motor\_t* \* *mp*, double *requested\_position* )

Move method for the fast shutter.

Slightly more complicated than a binary io as some flags need to be set up.

##### Parameters

<i>mp</i>	The fast shutter motor instance
<i>requested_position</i>	1 (open) or 0 (close), really

Definition at line 2517 of file lspmac.c.

```

    {
pthread_mutex_lock( &(mp->mutex));

mp->requested_position = requested_position;
mp->not_done = 1;
mp->motion_seen = 0;
mp->requested_pos_ctns = requested_position;
if( requested_position != 0) {
//
// ScanEnable=0, ManualEnable=1, ManualOn=1
//
lspmac_SockSendDLine( mp->name, "M1124=0 M1125=1
M1126=1");
} else {
//
// ManualOn=0, ManualEnable=0, ScanEnable=0
//
lspmac_SockSendDLine( mp->name, "M1126=0 M1125=0
M1124=0");
}

pthread_mutex_unlock( &(mp->mutex));

return 0;
}

```

#### 7.5.4.54 int lspmac\_moveabs\_queue ( ***lspmac\_motor\_t*** \* *mp*, double *requested\_position* )

Use coordinate system motion program, if available, to move motor to requested position.

##### Parameters

in	<i>mp</i>	The motor to move
in	<i>requested_position</i>	Where to move it

Definition at line 3472 of file Lspmac.c.

```

{
return lspmac_move_or_jog_abs_queue( mp,
requested_position, 0);
}
```

#### 7.5.4.55 void lspmac\_moveabs\_timed\_queue ( ***lspmac\_motor\_t*** \* *mp*, double *start*, double *delta*, double *time* )

timed motor move

##### Parameters

<i>mp</i>	Our motor object
<i>start</i>	Beginning of motion
<i>delta</i>	Distance to move
<i>time</i>	to move it in (secs)

< Flags needed for wait routine

Definition at line 2588 of file Lspmac.c.

```

{
// 240          LS-CAT Timed X move
//           Q10   = Starting X value (cnts)
//           Q11   = Delta X value   (cnts)
//           Q12   = Time to run between the two points (mSec)
//           Q13   = Acceleration time (msecs)
//           Q100  = 1 << (coord sys no - 1)

int q10;      // Starting value (counts)
```

```

int q11;      // Delta (counts)
int q12;      // Time to run (msecs)
int q13;      // Acceleration time (msecs)
int q100;     // 1 << (coord sys no - 1)
int coord_num; // our coordinate number
double u2c;
double neutral_pos;
double max_accel;

pthread_mutex_lock( &(mp->mutex));

u2c      = lsredis_getd( mp->u2c);
max_accel = lsredis_getd( mp->max_accel);
coord_num = lsredis_getl( mp->coord_num);
neutral_pos = lsredis_getd( mp->neutral_pos);

if( u2c == 0.0 || time <= 0.0 || max_accel <= 0.0) {
    //
    // Shouldn't try moving a motor that has bad motion parameters
    //
    pthread_mutex_unlock( &(mp->mutex));
    return;
}

mp->not_done    = 1;
mp->motion_seen = 0;

mp->requested_position = start + delta;
mp->requested_pos_cnts = u2c * (mp->requested_position
+ neutral_pos);
q10 = mp->requested_pos_cnts;
q11 = u2c * delta;
q12 = 1000 * time;
q13 = q11 / q12 / max_accel;
q100 = 1 << (coord_num - 1);
pthread_mutex_unlock( &(mp->mutex));

pthread_mutex_lock( &(mp->mutex));
lspmac_SockSendDPLine( mp->name, "&%d Q10=%d Q11=%d
Q12=%d Q13=%d Q100=%d B240R", coord_num, q10, q11, q12, q13, q100);
pthread_mutex_unlock( &(mp->mutex));
}

```

#### 7.5.4.56 int lsmpmac\_moveabs\_wait ( lsmpmac\_motor\_t \* mp, double timeout\_secs )

Wait for motor to finish moving.

Assume motion already queued, now just wait

##### Parameters

<i>mp</i>	The motor object to wait for
<i>timeout_secs</i>	The number of seconds to wait for. Fractional values fine.

Definition at line 3498 of file lsmpmac.c.

```

{
struct timespec timeout, now;
double isecs, fsecs;
int err;

//
// Copy the queue item for the most recent move request
//
clock_gettime( CLOCK_REALTIME, &now);

fsecs = modf( timeout_secs, &isecs);

timeout.tv_sec  = now.tv_sec + (long)floor( isecs);
timeout.tv_nsec = now.tv_nsec + (long)floor( fsecs * 1.0e9);

timeout.tv_sec  += timeout.tv_nsec / 1000000000;
timeout.tv_nsec %= 1000000000;

err = 0;
pthread_mutex_lock( &(mp->mutex));

while( err == 0 && mp->command_sent == 0)
    err = pthread_cond_timedwait( &mp->cond, &mp->mutex, &timeout);

```

```

pthread_mutex_unlock( &(mp->mutex));
if( err != 0) {
    if( err != ETIMEDOUT) {
        lslogging_log_message( "lspmac_moveabs_wait:
            unexpected error from timedwait %d tv_sec %ld tv_nsec %ld", err, timeout.tv_sec,
            timeout.tv_nsec);
    }
    return 1;
}

// 
// wait for the motion to have started
// This will time out if the motion ends before we can read the status back
// hence the added complication of time stamp of the sent packet.

err = 0;
pthread_mutex_lock( &(mp->mutex));
while( err == 0 && mp->motion_seen == 0)
    err = pthread_cond_timedwait( &(mp->cond), &(mp->mutex), &timeout)
;

if( err != 0) {
    if( err != ETIMEDOUT) {
        lslogging_log_message( "lspmac_moveabs_wait:
            unexpected error from timedwait: %d tv_sec %ld tv_nsec %ld", err, timeout.tv_sec,
            timeout.tv_nsec);
    }
    pthread_mutex_unlock( &(mp->mutex));
    return 1;
}

// 
// wait for the motion that we know has started to finish
// 
err = 0;
while( err == 0 && mp->not_done)
    err = pthread_cond_timedwait( &(mp->cond), &(mp->mutex), &timeout)
;

if( err != 0) {
    if( err != ETIMEDOUT) {
        lslogging_log_message( "lspmac_moveabs_wait:
            unexpected error from timedwait: %d tv_sec %ld tv_nsec %ld", err, timeout.tv_sec,
            timeout.tv_nsec);
    }
    pthread_mutex_unlock( &(mp->mutex));
    return 1;
}

// 
// if return code was not 0 then we know we shouldn't wait for not_done flag.
// In this case the motion ended before we read the status registers
//
pthread_mutex_unlock( &(mp->mutex));
return 0;
}

```

#### 7.5.4.57 int lspmac\_movedac\_queue ( *Lspmac\_motor\_t* \* *mp*, double *requested\_position* )

Move method for dac motor objects (ie, lights)

##### Parameters

in	<i>mp</i>	Our motor
in	<i>requested_position</i>	Desired x position (look up and send y position)

Definition at line 2383 of file *lspmac.c*.

```

{
double u2c;

pthread_mutex_lock( &(mp->mutex));

u2c = lsredis_getd( mp->u2c);
mp->requested_position = requested_position;

```

```

if( mp->nlut > 0 && mp->lut != NULL) {
    //
    // u2c scales the lookup table value
    //
    mp->requested_pos_cnts = u2c * lspmac_lut( mp->
        nlut, mp->lut, requested_position);

    lslogging_log_message( "lspmac_movedac_queue: motor %s
        requested position %f requested counts %d u2c %f",
        mp->name, mp->requested_position
        , mp->requested_pos_cnts, u2c);

    mp->not_done = 1;
    mp->motion_seen = 0;

    lspmac_SockSendDPLine( mp->name, "%s=%d", mp->
        dac_mvar, mp->requested_pos_cnts);
}

pthread_mutex_unlock( &(mp->mutex));
return 0;
}

```

#### 7.5.4.58 int lspmac\_movezoom\_queue ( Lspmac\_motor\_t \* mp, double requested\_position )

Move method for the zoom motor.

##### Parameters

in	<i>mp</i>	the zoom motor
in	<i>requested_position</i>	our desired zoom

Definition at line 2416 of file lspmac.c.

```

{
int motor_num;
int in_position_band;

lslogging_log_message( "lspmac_movezoom_queue: Here I am
    ");
pthread_mutex_lock( &(mp->mutex));

motor_num = lsredis_get1( mp->motor_num);
in_position_band = lsredis_get1( mp->in_position_band
    );

mp->requested_position = requested_position;

if( mp->nlut > 0 && mp->lut != NULL) {
    mp->requested_pos_cnts = lspmac_lut( mp->nlut
        , mp->lut, requested_position);

    if( abs( mp->requested_pos_cnts - mp->actual_pos_cnts
        ) * 16 <= in_position_band) {
        lslogging_log_message( "lspmac_movezoom_queue:
            Faking move");
        //
        // fake the move
        //
        mp->not_done = 1;
        mp->motion_seen = 0;
        mp->command_sent = 1;
        pthread_mutex_unlock( &(mp->mutex));
        lsevents_send_event( "%s Moving", mp->name);

        //
        // Perhaps give someone else a chance to process the move
        //
        pthread_mutex_lock( &(mp->mutex));
        mp->not_done = 0;
        mp->motion_seen = 1;
        mp->command_sent = 1;
        pthread_mutex_unlock( &(mp->mutex));
        lsevents_send_event( "%s In Position", mp->name);
        return 0;
    }
}

mp->not_done = 1;

```

```

    mp->motion_seen = 0;
    mp->command_sent = 0;

    lspmac_SockSendDPLine( mp->name, "#%d j=%d",
                           motor_num, mp->requested_pos_cnts);
}
pthread_mutex_unlock( &(mp->mutex));
lslogging_log_message( "lspmac_movezoom_queue: There you
                           were");
return 0;
}

```

#### 7.5.4.59 void lspmac\_next\_state( )

State machine logic.

Given the current state, generate the next one

Definition at line 2213 of file lspmac.c.

```

{
// Connect to the pmac and perhaps initialize it.
// OK, this is slightly more than just the state
// machine logic...
//
if( ls_pmac_state == LS_PMAC_STATE_DETACHED
) {
//
// TODO (eventually)
// This ip address wont change in a single PMAC installation
// We'll need to audit the code if we decide to implement
// multiple PMACs so might as well wait til then.
//
lsConnect( "192.6.94.5");

//
// If the connect was successful we can proceed with the initialization
//
if( ls_pmac_state != LS_PMAC_STATE_DETACHED
) {
    lspmac_SockFlush();

//
// Harvest the I and M variables in case we need them
// one day.
//
if( getmvars) {
    lspmac_GetAllMVars();
    getmvars = 0;
}

if( getivars) {
    lspmac_GetAllIVars();
    getivars = 0;
}
}

//
// Check the command queue and perhaps go to the "Send Command" state.
//
if( ls_pmac_state == LS_PMAC_STATE_IDLE) {
    int goodtogo;
    goodtogo = 0;
    pthread_mutex_lock( &lspmac_ascii_mutex);
    if( lspmac_ascii_busy==0 && lspmac_dpascii_on
        != lspmac_dpascii_off)
        goodtogo = 1;
    pthread_mutex_unlock( &lspmac_ascii_mutex);
    if( goodtogo)
        lspmac_SockSendDPqueue();
}

if( ls_pmac_state == LS_PMAC_STATE_IDLE &&
    ethCmdOn != ethCmdOff)
    ls_pmac_state = LS_PMAC_STATE_SC;
}

```

```

// Set the events flag
// to tell poll what we are waiting for.
//
switch( ls_pmac_state) {
case LS_PMAC_STATE_DETACHED:
//
// there shouldn't be a valid fd, so ignore the events
//
pmacfd.events = 0;
break;

case LS_PMAC_STATE_IDLE:
if( ethCmdOn == ethCmdOff) {
//
// Anytime we are idle we want to
// get the status of the PMAC
//
lspmac_get_status();
}

//
// These states require that we listen for packets
//
case LS_PMAC_STATE_WACK_NFR:
case LS_PMAC_STATE_WACK:
case LS_PMAC_STATE_WACK_CC:
case LS_PMAC_STATE_WACK_RR:
case LS_PMAC_STATE_WCR:
case LS_PMAC_STATE_WGB:
case LS_PMAC_STATE_GMR:
pmacfd.events = POLLIN;
break;

//
// These states require that we send packets out.
//
case LS_PMAC_STATE_SC:
case LS_PMAC_STATE_CR:
case LS_PMAC_STATE_RR:
case LS_PMAC_STATE_GB:
//
// Sad fact: PMAC will fail to process commands if we send them too
// quickly.
// We deal with that by waiting a tad before we let poll tell us the PMAC
// socket is ready to write.
//
gettimeofday( &now, NULL);
if( ((now.tv_sec * 1000000. + now.tv_usec) - (pmac_time_sent.tv_sec
* 1000000. + pmac_time_sent.tv_usec)) < PMAC_MIN_CMD_TIME) {
pmacfd.events = 0;
} else {
pmacfd.events = POLLOUT;
}
break;
}
}

```

#### 7.5.4.60 void lspmac\_pmacmotor\_read( lspmac\_motor\_t \* mp )

Read the position and status of a normal PMAC motor.

##### Parameters

in	<i>mp</i>	Our motor
----	-----------	-----------

Definition at line 1413 of file lspmac.c.

```

{
char s[512], *sp;
int homing1, homing2;
double u2c;
double neutral_pos;
int motor_num;
char *fmt;
int status_changed;
```

```

if( lsredis_getb( mp->active) != 1)
    return;

pthread_mutex_lock( &(mp->mutex));

//
// if this time and last time were both "in position"
// and the position changed significantly then log the event
//
// On E omega has been observed to change by 0x10000 on its own
// with no real motion.
//
if( mp->status2 & 1 && mp->status2 == *mp->status2_p
    && abs( mp->actual_pos_cnts - *mp->actual_pos_cnts_p
    ) > 256) {
    // lslogging_log_message( "Instantaneous change: %s old status1: %0x,
    new status1: %0x, old status2: %0x, new status2: %0x, old cnts: %0x, new cnts:
    %0x",
    // mp->name, mp->status1, *mp->status1_p, mp->status2,
    *mp->status2_p, mp->actual_pos_cnts, *mp->actual_pos_cnts_p);

    //
    // At this point we'll just log the event and return
    // There is no reason to believe the change is real.
    //
    // There is a non-zero probability that the first value is the bad one and
    any value afterwards will be taken as
    // wrong. Homing (or moving) the motor should fix this. There is a
    non-zero probably that it can happen
    // two or more times in a row after moving.
    //
    // TODO: account for the case where mp->actual_pos_cnts is the bad value.
    //
    // TODO: Is this a problem when the motor is moving? Can we detect it?
    //
    // TODO: Think of the correct change value here (currently 256) that works
    for all motors
    // or have this value configurable
    //
    pthread_mutex_unlock( &(mp->mutex));
    return;
}

// Send an event if inPosition has changed
//
if( (mp->status2 & 0x0000001) != (*mp->status2_p & 0x0000001))
{
    lsevents_send_event( "%s %s", mp->name, (*mp->
    status2_p & 0x0000001) ? "In Position" : "Moving");
}

// Get some values we might need later
//
u2c      = lsredis_getd( mp->u2c);
motor_num = lsredis_getl( mp->motor_num);
neutral_pos = lsredis_getd( mp->neutral_pos);

//
// maybe look for omega zero crossing
//
if( motor_num == 1 && omega_zero_search && *mp->
    actual_pos_cnts_p >=0 && mp->actual_pos_cnts <
    0) {
    int secs, nsecs;

    if( omega_zero_velocity > 0.0) {
        secs = *mp->actual_pos_cnts_p / omega_zero_velocity
        ;
        nsecs = (*mp->actual_pos_cnts_p / omega_zero_velocity
        - secs) * 1000000000;

        omega_zero_time.tv_sec = lspmactatus_time
        .tv_sec - secs;
        omega_zero_time.tv_nsec= lspmactatus_time
        .tv_nsec;
        if( omega_zero_time.tv_nsec < nsecs) {
            omega_zero_time.tv_sec -= 1;
            omega_zero_time.tv_nsec += 1000000000;
        }
        omega_zero_time.tv_nsec -= nsecs;
    }

    lsevents_send_event( "omega crossed zero");
    lslogging_log_message("lspmactpmacmotor_read: omega
    zero secs %d nsecs %d ozt.tv_sec %ld ozt.tv_nsec %ld, motor cnts %d",

```

```

        secs, nsecs, omega_zero_time.tv_sec,
        omega_zero_time.tv_nsec, *mp->actual_pos_cnts_p
    );
}
omega_zero_search = 0;
}

// Make local copies so we can inspect them in other threads
// without having to grab the status mutex
//
if( mp->status1 != *mp->status1_p || mp->status2 != *
    mp->status2_p ) {
    mp->status1 = *mp->status1_p;
    mp->status2 = *mp->status2_p;
    status_changed = 1;
} else {
    status_changed = 0;
}
mp->actual_pos_cnts = *mp->actual_pos_cnts_p;

//
// See if we are done moving, ie, in position
//
if( mp->status2 & 0x0000001) {
    if( mp->not_done) {
        mp->not_done = 0;
        pthread_cond_signal( &(mp->cond));
    }
} else if( mp->not_done == 0) {
    mp->not_done = 1;
}

// See if the motor is moving
//
//          move timer           homing
//          123456                 123456
if( mp->status1 & 0x020000 || mp->status1 & 0x0000400) {
    if( mp->motion_seen == 0) {
        mp->motion_seen = 1;
        pthread_cond_signal( &(mp->cond));
    }
}

pthread_mutex_lock( &ncurses_mutex);
mvwprintw( mp->win, 2, 1, "%*s", LS_DISPLAY_WINDOW_WIDTH
    -2, " ");
mvwprintw( mp->win, 2, 1, "%*d cts", LS_DISPLAY_WINDOW_WIDTH
    -6, mp->actual_pos_cnts);
mvwprintw( mp->win, 3, 1, "%*s", LS_DISPLAY_WINDOW_WIDTH
    -2, " ");
pthread_mutex_unlock( &ncurses_mutex);

if( mp->nlut >0 && mp->lut != NULL) {
    mp->position = lspmactrlut( mp->nlut, mp->lut, mp
        ->actual_pos_cnts);
} else {
    if( u2c != 0.0) {
        mp->position = ((mp->actual_pos_cnts / u2c) -
            neutral_pos);
    } else {
        mp->position = mp->actual_pos_cnts;
    }
}

if( status_changed || fabs(mp->reported_position - mp->
    position) >= lsredis_getd(mp->update_resolution
    )) {
    fmt = lsredis_getstr(mp->redis_fmt);
    lsredis_setstr( mp->redis_position, fmt, mp->
        position);
    free(fmt);
    mp->reported_position = mp->position;
}

fmt = lsredis_getstr( mp->printf_fmt);
snprintf( s, sizeof(s)-1, fmt, 8, mp->position);
s[sizeof(s)-1] = 0;
free( fmt);

//
// indicate limit problems
//
lsredis_setstr( mp->pos_limit_hit, mp->status1
    & 0x200000 ? "1" : "0");
lsredis_setstr( mp->neg_limit_hit, mp->status1
    & 0x400000 ? "1" : "0");

```

```

// set flag if we are not homed
homing1 = 0;
//                                     ~ (hommed flag)
if( mp->homing == 0 && (~mp->status2 & 0x000400) != 0) {
    homing1 = 1;
}

// set flag if we are homing and in open loop
homing2 = 0;
//                                     open loop
if( mp->homing == 1 && (mp->status1 & 0x040000) != 0) {
    homing2 = 1;
}

// maybe reset homing flag
//                                     homed flag                                in position flag
if( (mp->homing == 2) && ((mp->status2 & 0x000400) != 0) && ((mp
->status2 & 0x000001) != 0)) {
    mp->homing = 0;
    lsevents_send_event( "%s Homed", mp->name);
}

pthread_mutex_lock( &ncurses_mutex);
s[sizeof(s)-1] = 0;
mvwprintw( mp->win, 3, 1, "%*s", LS_DISPLAY_WINDOW_WIDTH
-6, s);

if( status_changed) {
    mvwprintw( mp->win, 4, 1, "%*x", LS_DISPLAY_WINDOW_WIDTH
-2, mp->status1);
    mvwprintw( mp->win, 5, 1, "%*x", LS_DISPLAY_WINDOW_WIDTH
-2, mp->status2);
    sp = "";
    if( mp->status2 & 0x000002)
        sp = "Following Warning";
    else if( mp->status2 & 0x000004)
        sp = "Following Error";
    else if( mp->status2 & 0x000020)
        sp = "I2T Amp Fault";
    else if( mp->status2 & 0x000008)
        sp = "Amp. Fault";
    else if( mp->status2 & 0x000080)
        sp = "Stopped on Limit";
    else if( mp->status1 & 0x040000)
        sp = "Open Loop";
    else if( ~(mp->status1) & 0x080000)
        sp = "Motor Disabled";
    else if( mp->status1 & 0x000400)
        sp = "Homing";
    else if( (mp->status1 & 0x600000) == 0x600000)
        sp = "Both Limits Tripped";
    else if( mp->status1 & 0x200000)
        sp = "Positive Limit";
    else if( mp->status1 & 0x400000)
        sp = "Negative Limit";
    else if( ~(mp->status2) & 0x000400)
        sp = "Not Homed";
    else if( mp->status1 & 0x020000)
        sp = "Moving";
    else if( mp->status2 & 0x000001)
        sp = "In Position";

    mvwprintw( mp->win, 6, 1, "%*s", LS_DISPLAY_WINDOW_WIDTH
-2, sp);

    lsredis_setstr( mp->status_str, sp);
}
wnoutrefresh( mp->win);
pthread_mutex_unlock( &ncurses_mutex);

pthread_mutex_unlock( &(mp->mutex));

if( homing1)
    lspmac_home1_queue( mp);

if( homing2)
    lspmac_home2_queue( mp);

lspmac_status_last_time.tv_sec = lspmac_status_time
    .tv_sec;
lspmac_status_last_time.tv_nsec = lspmac_status_time
    .tv_nsec;
}

```

## 7.5.4.61 pmac\_cmd\_queue\_t\* lspmac\_pop\_queue( )

Remove the oldest queue item.

Used to send command to PMAC. Note that there is a separate reply index to ensure we've know to what command a reply is referring. Returns the item.

Definition at line 671 of file lspmac.c.

```
{
pmac_cmd_queue_t *rtn;

pthread_mutex_lock( &pmac_queue_mutex);

if( ethCmdOn == ethCmdOff)
    rtn = NULL;
else {
    rtn = &(ethCmdQueue[(ethCmdOff++) %
    PMAC_CMD_QUEUE_LENGTH]);
    clock_gettime( CLOCK_REALTIME, &(rtn->time_sent));
}
pthread_mutex_unlock( &pmac_queue_mutex);
return rtn;
}
```

## 7.5.4.62 pmac\_cmd\_queue\_t\* lspmac\_pop\_reply( )

Remove the next command queue item that is waiting for a reply.

We always need a reply to know we are done with a given command. Returns the item.

Definition at line 691 of file lspmac.c.

```
{
pmac_cmd_queue_t *rtn;

pthread_mutex_lock( &pmac_queue_mutex);

if( ethCmdOn == ethCmdReply)
    rtn = NULL;
else
    rtn = &(ethCmdQueue[(ethCmdReply++) %
    PMAC_CMD_QUEUE_LENGTH]);

pthread_mutex_unlock( &pmac_queue_mutex);
return rtn;
}
```

## 7.5.4.63 pmac\_cmd\_queue\_t\* lspmac\_push\_queue( pmac\_cmd\_queue\_t \*cmd )

Put a new command on the queue.

Pointer is returned so caller can evaluate the time command was actually sent.

## Parameters

<i>cmd</i>	Command to send to the PMAC
------------	-----------------------------

Definition at line 647 of file lspmac.c.

```
{
pmac_cmd_queue_t *rtn;

pthread_mutex_lock( &pmac_queue_mutex);
rtn = &(ethCmdQueue[(ethCmdOn++) % PMAC_CMD_QUEUE_LENGTH
]);
memcpy( rtn, cmd, sizeof( pmac_cmd_queue_t));
rtn->time_sent.tv_sec = 0;
rtn->time_sent.tv_nsec = 0;
pthread_cond_signal( &pmac_queue_cond);
```

```

pthread_mutex_unlock( &pmac_queue_mutex);

return rtn;
}

```

#### 7.5.4.64 void lsmpmac\_quitting\_cb ( char \* event )

prepare to exit program in a couple of seconds

Definition at line 4088 of file lsmpmac.c.

```

{
double move_time;
int mmask;

pgpmac_request_stay_of_execution( 1 );
fshut->moveAbs( fshut, 0.0 );
dryer->moveAbs( dryer, 0.0 );

lsmpmac_est_move_time( &move_time, &mmask,
    aperz, 1, "Cover", 0.0,
    capz, 1, "Cover", 0.0,
    scint, 1, "Cover", 0.0,
    blight, 1, NULL, 0.0,
    flight, 1, NULL, 0.0,
    NULL);

pgpmac_request_stay_of_execution( ((int)
    move_time) + 2 );

}

```

#### 7.5.4.65 void lsmpmac\_request\_control\_response\_cb ( char \* event )

Definition at line 2044 of file lsmpmac.c.

```

{
static char s[32];
int i;

for( i=0; i<31 && event[i] != 0; i++ ) {
    s[i] = 0;
    if( event[i] == ' ' )
        break;
    s[i] = event[i];
}
s[i] = 0;
lsmpmac_get_ascii( s );

}

```

#### 7.5.4.66 void lsmpmac\_Reset ( )

Clear the queue and put the PMAC into a known state.

Definition at line 770 of file lsmpmac.c.

```

{
ls_pmac_state = LS_PMAC_STATE_IDLE;

// clear queue
ethCmdReply = ethCmdOn;
ethCmdOff = ethCmdOn;

lsmpmac_SockFlush();
}

```

## 7.5.4.67 void lspmac\_reset\_queue( )

Clear the queue as part of PMAC reinitialization.

Definition at line 634 of file lspmac.c.

```
{
pthread_mutex_lock( &pmac_queue_mutex );
ethCmdOn    = 0;
ethCmdOff   = 0;
ethCmdReply = 0;
pthread_mutex_unlock( &pmac_queue_mutex );
}
```

## 7.5.4.68 double lspmac\_rlut( int nlut, double \* lut, double y )

## Parameters

in	<i>nlut</i>	number of entries in lookup table
in	<i>lut</i>	our lookup table
in	<i>y</i>	the y value for which we need an x

Definition at line 455 of file lspmac.c.

```
{
int i, foundone, up;
double m;
double y1, y2, x1, x2, x;

foundone = 0;
if( lut != NULL && nlut > 1) {

    //
    // are the table values going up or down?
    //
    if( lut[1] < lut[2*nlut-1])
        up = 1;
    else
        up = 0;

    //
    // Linear search
    //
    for( i=0; i < 2*nlut; i += 2) {
        x1 = lut[i];
        y1 = lut[i+1];
        if( i < 2*nlut - 2) {
            x2 = lut[i+2];
            y2 = lut[i+3];
        }
        //
        // see if y is before the beginning of the table
        //
        if( i==0 && ( up ? y1 > y : y1 < y)) {
            x = x1;
            foundone = 1;
            break;
        }
        //
        // Did we, perhaps, nail it?
        //
        if( y1 == y) {
            x = x1;
            foundone = 1;
            break;
        }
        //
        // Interpolate between the two values (if we've not bumped our heads on
        // the end of the table)
        //
        if( (i < 2*nlut-2) && (up ? y < y2 : y > y2)) {
            m = (x2 - x1) / (y2 - y1);
            x = m * (y - y1) + x1;
            foundone = 1;
            break;
        }
    }
}
```

```

// y is off the charts: just use the last value
//
if( foundone == 0 ) {
    x = lut[2*(nlut-1)];
}
return x;
}
return 0.0;
}

```

#### 7.5.4.69 void lspmac\_run( )

Start up the lspmac thread.

Definition at line 4359 of file lspmac.c.

```

{
static int first_time = 1;
char **inits;
lspmac_motor_t *mp;
char evts[64];
int i;
int active;
int motor_num;

pthread_create( &pmac_thread, NULL, lspmac_worker,
                NULL);

if( first_time) {
    first_time = 0;
    lsevents_add_listener( "^CryoSwitchChanged$",
                           lspmac_cryoSwitchChanged_cb);
    lsevents_add_listener( "^scint In Position$",
                           lspmac_scint_maybe_turn_on_dryer_cb);
    lsevents_add_listener( "^scint Moving$",
                           lspmac_scint_maybe_turn_off_dryer_cb);
    lsevents_add_listener( "^scintDried$",
                           lspmac_scint_dried_cb);
    lsevents_add_listener( "^backLight 1$",
                           lspmac_backLight_up_cb);
    lsevents_add_listener( "^backLight 0$",
                           lspmac_backLight_down_cb);
    lsevents_add_listener( "^cam.zoom Moving$",
                           lspmac_light_zoom_cb);
//    lsevents_add_listener( "^Quitting Program$",
                           lspmac_quitting_cb);
    lsevents_add_listener( "^Control-[BCFGV] accepted$",
                           lspmac_request_control_response_cb);
    lsevents_add_listener( "^Full Card Reset$",
                           lspmac_full_card_reset_cb);

    if( pgpmac_use_automscint) {
        lsevents_add_listener( "~scint In Position$",
                               lspmac_scint_maybe_return_sample_cb);
        lsevents_add_listener( "^scint Moving$",
                               lspmac_scint_maybe_move_sample_cb);
    }

    for( i=0; i<lspmac_nmotors; i++) {
        snprintf( evts, sizeof( evts)-1, "%s command accepted$",
                  lspmac_motors[i].name);
        evts[sizeof(evts)-1] = 0;
        lsevents_add_listener( evts, lspmac_command_done_cb
        );
    }

    lspmac_zoom_lut_setup();
    lspmac_flight_lut_setup();
    lspmac_blight_lut_setup();
    lspmac_fscint_lut_setup();
}

// Clear the command interfaces
//
// lspmac_SockSendControlCharPrint( "Control-X", '\x18'); // why does this
// kill the initialization?

```

```

{
    uint32_t cc;
    cc = 0;
    lspmacc_send_command( VR_UPLOAD, VR_PMAC_SETMEM
        , 0x0e9e, 0, 4, (char *)&cc, NULL, 1, NULL);

    cc = 0x18;
    lspmacc_send_command( VR_UPLOAD, VR_PMAC_SETMEM
        , 0x0e9e, 0, 4, (char *)&cc, NULL, 1, NULL);
}
// Initialize the MD2 pmac (ie, turn on the right plcc's etc)
// /*
for( inits = lsredis_get_string_array(lspmacc_md2_init); *inits != NULL;
    inits++) {
    lspmacc_SockSendDPLine( NULL, *inits);
}
*/
// Initialize the pmac's support for each motor
// (ie, set the various flag for when a motor is active or not)
// /*
for( i=0; i<lspmacc_nmotors; i++) {
    mp          = &(lspmacc_motors[i]);
    active      = lsredis_getb( mp->active);
    motor_num   = lsredis_getl( mp->motor_num);

    if( motor_num >= 1 && motor_num <= 32) {

        // Set the PMAC to be consistant with redis
        // lspmacc_SockSendDPLine( NULL, "I%d16=%f I%d17=%f
        // I%d28=%d", motor_num, lsredis_getd( mp->max_speed), motor_num,
        lsredis_getd( mp->max_accel), motor_num, lsredis_getl
        ( mp->in_position_band));
    }
}

// if there is a problem with "active" then don't do anything
// On the other hand, various combinations of yes/no true/false 1/0 should
// work
// switch( active) {
case 1:
    inits = lsredis_get_string_array( mp->active_init
    );
    break;

case 0:
    inits = lsredis_get_string_array( mp->
    inactive_init);
    break;

default:
    lslogging_log_message( "lspmacc_run: motor %s is
        neither active nor inactive (!?)", mp->name);
    inits = NULL;
}
if( inits != NULL) {
    while( *inits != NULL) {
        lspmacc_SockSendDPLine( NULL, *inits);
        inits++;
    }
}
}

```

**7.5.4.70 void lspmac\_scint\_dried\_cb ( char \* event )**

Turn off the dryer.

## Parameters

*event* required by protocol

Definition at line 4176 of file `Ispmac.c`.

{

```

lslogging_log_message( "lspmac_scint_dried_cb: Stopping
    dryer");
dryer->moveAbs( dryer, 0.0);
}

```

#### 7.5.4.71 void lspmac\_scint\_maybe\_move\_sample\_cb ( char \* event )

Perhaps we need to move the sample out of the way.

Definition at line 4110 of file lspmac.c.

```

{
static int trigger = 1;
double scint_target;
int err;
double move_time;
int mmask;

pthread_mutex_lock( &scint->mutex);
scint_target = scint->requested_position;
pthread_mutex_unlock( &scint->mutex);

// This should be pretty conservative since the out position is around 80
//
if( scint_target > 10.0) {
    if( trigger) {
        mmask = 0;
        err = lspmac_est_move_time( &move_time, &mmask,
                                    alignx, 0, "Back", -2.0,
                                    aligny, 0, "Back", 1.0,
                                    alignz, 0, "Back", 1.0,
                                    NULL);
        if( err) {
            lspmac_abort();
            lsevents_send_event( "Move Aborted");
            lslogging_log_message( "lspmac_scint_maybe_move_sample_cb: Failed move request, aborting motion to keep scint from hitting sample");
        }
        trigger = 0;
    }
} else {
    trigger = 1;
}
}

```

#### 7.5.4.72 void lspmac\_scint\_maybe\_return\_sample\_cb ( char \* event )

Perhaps we need to return the sample to the beam.

Definition at line 4145 of file lspmac.c.

```

{
static int trigger = 1;
double scint_target;
double move_time;
int mmask;

pthread_mutex_lock( &scint->mutex);
scint_target = scint->requested_position;
pthread_mutex_unlock( &scint->mutex);

// This should be pretty conservative since the out position is around 80
//
if( scint_target < 10.0) {
    if( trigger) {
        mmask = 0;
        lspmac_est_move_time( &move_time, &mmask,
                            alignx, 0, "Beam", 0.0,
                            aligny, 0, "Beam", 0.0,
                            alignz, 0, "Beam", 0.0,
                            NULL);
        trigger = 0;
    }
} else {
    trigger = 1;
}
}

```

## 7.5.4.73 void lspmac\_scint\_maybe\_turn\_off\_dryer\_cb ( char \* event )

Maybe stop drying off the scintilator.

## Parameters

<code>event</code>	required by protocol
--------------------	----------------------

Definition at line 4031 of file lspmac.c.

```
double pos;

// See if the dryer is on
//
pos = lspmac_getPosition( dryer);

if( pos == 0.0)
    return;

dryer->moveAbs( dryer, 0.0);

lstimer_unset_timer( "scintDried");

}
```

## 7.5.4.74 void lspmac\_scint\_maybe\_turn\_on\_dryer\_cb ( char \* event )

Maybe start drying off the scintilator.

## Parameters

<code>event</code>	required by protocol
--------------------	----------------------

Definition at line 3994 of file lspmac.c.

```
{

static int trigger = 0;
double pos;
double cover;
int err;

pthread_mutex_lock( &(scint->mutex));
pos = scint->position;
pthread_mutex_unlock( &(scint->mutex));

if( pos > 20.0) {
    trigger = 1;
    return;
}

if( trigger == 0) {
    return;
}

err = lsredis_find_preset( scint->name, "Cover",
    &cover);

lslogging_log_message( "lspmac_scint_inPosition_cb: pos
    %f, cover %f, diff %f, err %d", pos, cover, fabs( pos-cover), err);

if( err == 0)
    return;

if( fabs( pos - cover) <= 0.1) {
    dryer->moveAbs( dryer, 1.0);
    lslogging_log_message( "lspmac_scint_inPosition_cb:
        Starting dryer");
    lstimer_set_timer( "scintDried", 1, 120, 0);
    trigger = 0;
}
}
```

**7.5.4.75 `pmac_cmd_queue_t* lspmacc_send_command( int rqType, int rq, int wValue, int wIndex, int wLength, char * data, void(*)(pmac_cmd_queue_t *, int, char *) responseCB, int no_reply, char * event )`**

Compose a packet and send it to the PMAC.

This is the meat of the PMAC communications routines. The queued command is returned.

#### Parameters

in	<i>rqType</i>	VR_UPLOAD or VR_DOWNLOAD
in	<i>rq</i>	PMAC command (see PMAC User Manual)
in	<i>wValue</i>	Command argument 1
in	<i>wIndex</i>	Command argument 2
in	<i>wLength</i>	Length of data array
in	<i>data</i>	Data array (or NULL)
in	<i>responseCB</i>	Function to call when a response is read from the PMAC
in	<i>no_reply</i>	Flag, non-zero means no reply is expected
in	<i>event</i>	base name for events

Definition at line 709 of file lspmacc.c.

```

{
static pmac_cmd_queue_t cmd;

cmd.pcmd.RequestType = rqType;
cmd.pcmd.Request = rq;
cmd.pcmd.wValue = htons(wValue);
cmd.pcmd.wIndex = htons(wIndex);
cmd.pcmd.wLength = htons(wLength);
cmd.onResponse = responseCB;
cmd.no_reply = no_reply;
cmd.event = event;

//
// Setting the message buff bData requires a bit more care to avoid over
// filling it
// or sending garbage in the unused bytes.
//

if( wLength > sizeof( cmd.pcmd.bData) ) {
    //
    // Bad things happen if we do not catch this case.
    //
    lslogging_log_message( "Message Length %d longer than
        maximum of %ld, aborting", wLength, sizeof( cmd.pcmd.bData));
    exit( -1);
}
if( data == NULL ) {
    memset( cmd.pcmd.bData, 0, sizeof( cmd.pcmd.bData));
} else {
    //
    // This could leave bData non-null terminated. I do not know if this is a
    // problem.
    //
    if( wLength > 0 )
        memcpy( cmd.pcmd.bData, data, wLength);
    if( wLength < sizeof( cmd.pcmd.bData) )
        memset( cmd.pcmd.bData + wLength, 0, sizeof( cmd.pcmd.bData)
            - wLength);
}
return lspmacc_push_queue( &cmd);
}

```

**7.5.4.76 `void lspmacc_sendcmd( char * event, void(*)(pmac_cmd_queue_t *, int, char *) responseCB, char * fmt, ... )`**

PMAC command with call back.

#### Parameters

in	<i>event</i>	base name for events
in	<i>responseCB</i>	our callback routine
in	<i>fmt</i>	printf style format string

Definition at line 2192 of file lspmac.c.

```

    {
static char tmps[1024];
va_list arg_ptr;

va_start( arg_ptr, fmt);
vsnprintf( tmps, sizeof(tmps)-1, fmt, arg_ptr);
tmps[sizeof(tmps)-1]=0;
va_end( arg_ptr);

lspmac_send_command( VR_DOWNLOAD,
VR_PMAC_SENDLINE, 0, 0, strlen(tmps), tmps, responseCB, 0,
event);
}

```

#### 7.5.4.77 void lspmac\_sendcmd\_noob ( char \* fmt, ... )

Send a command that does not need to deal with the reply.

##### Parameters

in	<i>fmt</i>	A printf style format string
----	------------	------------------------------

Definition at line 2173 of file lspmac.c.

```

    {
static char tmps[1024];
va_list arg_ptr;

va_start( arg_ptr, fmt);
vsnprintf( tmps, sizeof(tmps)-1, fmt, arg_ptr);
tmps[sizeof(tmps)-1]=0;
va_end( arg_ptr);

lspmac_send_command( VR_DOWNLOAD,
VR_PMAC_SENDLINE, 0, 0, strlen(tmps), tmps, NULL, 0, NULL);
}

```

#### 7.5.4.78 void lspmac\_SendControlReplyPrintCB ( pmac\_cmd\_queue\_t \* cmd, int nreceived, char \* buff )

Receive a reply to a control character Print a "printable" version of the character to the terminal Followed by a hex dump of the response.

##### Parameters

in	<i>cmd</i>	Queue item this is a reply to
in	<i>nreceived</i>	Number of bytes received
in	<i>buff</i>	Buffer of bytes received

Definition at line 1039 of file lspmac.c.

```

    {

char *sp;
int i;

sp = calloc( nreceived+1, 1);
for( i=0; i<nreceived; i++) {
    if( buff[i] == 0)
        break;
    if( isascii(buff[i]) && !iscntrl(buff[i]))
        sp[i] = buff[i];
    else
        sp[i] = ' ';
}
sp[i] = 0;

lslogging_log_message( "control-%c: %s", '@'+ ntohs(cmd

```

```

    ->pcmd.wValue), sp);
    free( sp);
}

```

#### 7.5.4.79 void lsmpmac\_Service ( struct pollfd \* evt )

Service routine for packet coming from the PMAC.

All communications is asynchronous so this is the only place incomming packets are handled

##### Parameters

in	evt	pollfd object returned by poll
----	-----	--------------------------------

Definition at line 810 of file lsmpmac.c.

```

{
static char *receiveBuffer = NULL;      // the buffer inwhich to stick our
    incoming characters
static int receiveBufferSize = 0;        // size of receiveBuffer
static int receiveBufferIn = 0;          // next location to write to in
    receiveBuffer
pmac_cmd_queue_t *cmd;                // maybe the
    command we are servicing
ssize_t nsent, nread;                 // nbytes dealt with
int i;                                // loop counter
int foundEOCR;                        // end of command response flag

if( evt->revents & (POLLERR | POLLHUP | POLLNVAL)) {
    if( evt->fd != -1) {
        close( evt->fd);
        evt->fd = -1;
    }
    ls_pmac_state = LS_PMAC_STATE_DETACHED;
    return;
}

if( evt->revents & POLLOUT) {

switch( ls_pmac_state) {
case LS_PMAC_STATE_DETACHED:
    break;
case LS_PMAC_STATE_IDLE:
    break;

case LS_PMAC_STATE_SC:
    cmd = lsmpmac_pop_queue();
    if( cmd == NULL)
        return;

    if( cmd->pcmd.Request == VR_PMAC_GETMEM) {
        nsent = send( evt->fd, cmd, pmac_cmd_size, 0);
        if( nsent != pmac_cmd_size) {
            lslogging_log_message( "Could only send %d of %d
bytes....Not good.", (int)nsent, (int)(pmac_cmd_size));
        }
    } else {
        nsent = send( evt->fd, cmd, pmac_cmd_size + ntohs(cmd->
pcmd.wLength), 0);
        gettimeofday( &pmac_time_sent, NULL);
        if( nsent != pmac_cmd_size + ntohs(cmd->pcmd.wLength)
)) {
            lslogging_log_message( "Could only send %d of %d
bytes....Not good.", (int)nsent, (int)(pmac_cmd_size + ntohs(cmd->
pcmd.wLength)));
        }
    }
}

if( cmd->pcmd.Request == VR_PMAC_SENDCTRLCHAR
)
    ls_pmac_state = LS_PMAC_STATE_WACK_CC
;
else if( cmd->pcmd.Request == VR_CTRL_RESPONSE
)
    ls_pmac_state = LS_PMAC_STATE_IDLE;
else if( cmd->pcmd.Request == VR_PMAC_GETMEM)
    ls_pmac_state = LS_PMAC_STATE_GMR;
else if( cmd->no_reply == 0)

```



```

        receiveBuffer[i] = 0;
        break;
    }
}

receiveBufferIn = i;
}

cmd = NULL;

switch( ls_pmac_state) {
case LS_PMAC_STATE_WACK_NFR:
    receiveBuffer[--receiveBufferIn] = 0;
    cmd = lspmac_pop_reply();
    ls_pmac_state = LS_PMAC_STATE_IDLE;
    break;
case LS_PMAC_STATE_WACK:
    receiveBuffer[--receiveBufferIn] = 0;
    ls_pmac_state = LS_PMAC_STATE_RR;
    break;
case LS_PMAC_STATE_WACK_CC:
    receiveBuffer[--receiveBufferIn] = 0;
    ls_pmac_state = LS_PMAC_STATE_CR;
    break;
case LS_PMAC_STATE_WACK_RR:
    receiveBufferIn -= 2;
    if( receiveBuffer[receiveBufferIn])
        ls_pmac_state = LS_PMAC_STATE_GB;
    else
        ls_pmac_state = LS_PMAC_STATE_RR;
    receiveBuffer[receiveBufferIn] = 0;
    break;
case LS_PMAC_STATE_GMR:
    cmd = lspmactl_pop_reply();
    ls_pmac_state = LS_PMAC_STATE_IDLE;
    break;
case LS_PMAC_STATE_WCR:
    cmd = lspmactl_pop_reply();
    ls_pmac_state = LS_PMAC_STATE_IDLE;
    break;
case LS_PMAC_STATE_WGB:
    if( foundEOCR) {
        cmd = lspmactl_pop_reply();
        ls_pmac_state = LS_PMAC_STATE_IDLE;
    } else {
        ls_pmac_state = LS_PMAC_STATE_RR;
    }
    break;
}

if( cmd != NULL && cmd->onResponse != NULL) {
    cmd->onResponse( cmd, receiveBufferIn, receiveBuffer);
    receiveBufferIn = 0;
}
}

```

7.5.4.80 int lsppmac\_set\_motion\_flags ( int \* mmaskp, lsppmac\_motor\_t \* mp\_1, ... )

Set the coordinate system motion flags (m5075) for the null terminated list of motors that we are planning on running a motion program with.

Note that `lspmac_est_move_time` already takes care of this, use when calling a motion program directly

## Parameters

<i>mmaskp</i>	Returned value of the mask generated. Ignored if null.
<i>mp_1</i>	start of null terminated list of motors.

Definition at line 2743 of file `Ispmac.c`.

```
va_list arg_ptr;
struct timespec timeout;
int err;
int cn;
int need_flag;
```

```

lspmacc_motor_t *mp;
int mmask;

mmask = 0;
if( mmaskp != NULL)
    *mmaskp = 0;

if( mp_1==NULL)
    return 0;

// add the coordinate system flags to mmask
// va_start( arg_ptr, mp_1);
for( mp = mp_1; mp!=NULL; mp = va_arg( arg_ptr, lspmacc_motor_t
    *)) {
    if( mp->magic != LSMPMAC_MAGIC_NUMBER) {
        lslogging_log_message( "lspmacc_set_motion_flags:
            WARNING: motor list must be NULL terminated. Check your call to
            lspmacc_set_motion_flags.");
        break;
    }
    cn = lsredis_getl( mp->coord_num);
    if( cn < 1 || cn > 16)
        continue;

    mmask |= 1 << (cn - 1);
}
va_end( arg_ptr);

if( mmaskp != NULL)
    *mmaskp = mmask;

// It could be the flag is already what we want. We might set up a race
// condition if we
// try to set it again. (so don't)
// pthread_mutex_lock( &lspmacc_moving_mutex);

if( (lspmacc_moving_flags & mmask) != 0)
    need_flag = 0;
else
    need_flag = 1;

pthread_mutex_unlock( &lspmacc_moving_mutex);

if( !need_flag)
    return 0;

// Set m5075 and make sure it propagates
// lspmacc_SockSendDPLine( NULL, "M5075=(M5075 | %d)", mmask
    );
clock_gettime( CLOCK_REALTIME, &timeout);
timeout.tv_sec += 2;

err = 0;
pthread_mutex_lock( &lspmacc_moving_mutex);
while( err == 0 && (lspmacc_moving_flags & mmask) != mmask)
    err = pthread_cond_timedwait( &lspmacc_moving_cond, &
        lspmacc_moving_mutex, &timeout);

pthread_mutex_unlock( &lspmacc_moving_mutex);

if( err == ETIMEDOUT) {
    lslogging_log_message( "lspmacc_set_motion_flags: timed
        out waiting for motion %d flag to be set", mmask);
    return 1;
}
return 0;
}

```

#### 7.5.4.81 void lspmacc\_shutter\_read ( lspmacc\_motor\_t \* mp )

Fast shutter read routine The shutter is mildly complicated in that we need to take into account the fact that the shutter can open and close again between status updates.

This means that we need to rely on a PCL program running in the PMAC to monitor the shutter state and let us know that this has happened.

**Parameters**

in	<i>mp</i>	The motor object associated with the fast shutter
----	-----------	---

Definition at line 1228 of file lspmacc.c.

```

{
char *fmt;
// track the shutter state and signal if it has changed
//
pthread_mutex_lock( &lspmacc_shutter_mutex );
if( md2_status.fs_has_opened && !
    lspmacc_shutter_has_opened && !md2_status.
    fs_is_open) {
//
// Here the shutter opened and closed again before we got the memo
// Treat it as a shutter closed event
//
pthread_cond_signal( &lspmacc_shutter_cond );
}
lspmacc_shutter_has_opened = md2_status.
    fs_has_opened;

if( lspmacc_shutter_state != md2_status.
    fs_is_open) {
    lspmacc_shutter_state = md2_status.fs_is_open
    ;
    pthread_cond_signal( &lspmacc_shutter_cond );
}

pthread_mutex_lock( &ncurses_mutex );
if( md2_status.fs_is_open) {
    mvwprintw( term_status2, 1, 1, "Shutter Open  ");
    mp->position = 1;
} else {
    mvwprintw( term_status2, 1, 1, "Shutter Closed");
    mp->position = 0;
}
pthread_mutex_unlock( &ncurses_mutex );

if( fshut->reported_position != fshut->position
    ) {
    fmt = lsredis_getstr( fshut->redis_fmt);
    lsredis_setstr( fshut->redis_position, fmt
        , fshut->position);
    free(fmt);
    fshut->reported_position = fshut->position
    ;
}
pthread_mutex_unlock( &lspmacc_shutter_mutex );
}

```

**7.5.4.82 void lspmacc\_SockFlush( )**

Reset the PMAC socket from the PMAC side.

Puts the PMAC into a known communications state

Definition at line 763 of file lspmacc.c.

```

{
lspmacc_send_command( VR_DOWNLOAD, VR_PMAC_FLUSH
    , 0, 0, 0, NULL, NULL, 1, NULL);
}

```

**7.5.4.83 pmac\_cmd\_queue\_t\* lspmacc\_SockGetmem( int offset, int nbytes )**

Request a chunk of memory to be returned.

**Parameters**

in	<i>offset</i>	Offset in PMAC Double Buffer
in	<i>nbytes</i>	Number of bytes to request

Definition at line 1082 of file lspmac.c.

```

    {
    return lspmac_send_command( VR_UPLOAD,
        VR_PMAC_GETMEM, offset, 0, nbytes, NULL, lspmac_GetmemReplyCB
        , 0, NULL);
}

```

#### 7.5.4.84 pmac\_cmd\_queue\_t\* lspmac\_SockSendControlCharPrint ( char \* event, char c )

Send a control character.

##### Parameters

in	<i>event</i>	base name for events
	<i>c</i>	The control character to send

Definition at line 1135 of file lspmac.c.

```

    {
lspmac_control_char = c;
//  return lspmac_send_command( VR_DOWNLOAD, VR_PMAC_SENDCTRLCHAR, c, 0,
//      1400, NULL, lspmac_SendControlReplyPrintCB, 0, event);
return lspmac_send_command( VR_UPLOAD,
    VR_CTRL_RESPONSE, c, 0, 1400, NULL,
    lspmac_SendControlReplyPrintCB, 0, event);
}

```

#### 7.5.4.85 void lspmac\_SockSendDPControlChar ( char \* event, char c )

use dpram ascii interface to send a control character

Definition at line 2067 of file lspmac.c.

```

    {
uint16_t buff;

buff = 0x07 & c;
lspmac_send_command( VR_UPLOAD, VR_PMAC_SETMEM
    , 0x0e9e, 0, 2, (char *)&buff, lspmac_SockSendDPControlCharCB
    , 1, event);
if( event != NULL)
lsevents_send_event( "%s queued", event);
}

```

#### 7.5.4.86 void lspmac\_SockSendDPControlCharCB ( pmac\_cmd\_queue\_t \* cmd, int nreceived, char \* buf )

Definition at line 2060 of file lspmac.c.

```

    {
if( cmd->event != NULL && *(cmd->event))
lsevents_send_event( "%s accepted", cmd->event);
}

```

#### 7.5.4.87 void lspmac\_SockSendDpline ( char \* event, char \* fmt, ... )

prepare (queue up) a line to send the dpram ascii command interface

Definition at line 2024 of file lspmac.c.

```

    {
va_list arg_ptr;
uint32_t index;
char *p1;

pthread_mutex_lock( &lspmac_ascii_mutex);
index = lspmac_dpascii_on++ % LSPMAC_DPASCII_QUEUE_LENGTH
;

p1 = lspmac_dpascii_queue[index].p1;

va_start( arg_ptr, fmt);
vsnprintf( p1, 159, fmt, arg_ptr);
p1[159] = 0;
va_end( arg_ptr);

lspmac_dpascii_queue[index].event = event;
pthread_mutex_unlock( &lspmac_ascii_mutex);
}

```

#### 7.5.4.88 void lspmac\_SockSendDPqueue( )

Definition at line 2077 of file lspmac.c.

```

{
lspmac_dpascii_queue_t *qp;
uint32_t mask;
uint32_t clrdata;

pthread_mutex_lock( &lspmac_ascii_mutex);
qp = &(lspmac_dpascii_queue[(lspmac_dpascii_off
    ++) % LSPMAC_DPASCII_QUEUE_LENGTH]);
lspmac_ascii_busy = 1;
pthread_mutex_unlock( &lspmac_ascii_mutex);

lslogging_log_message( "lspmac_SockSendDPqueue: %s", qp
    ->p1);

clrdata = 0;           // set the control word to zero
lspmac_send_command( VR_UPLOAD, VR_PMAC_SETMEM
    , 0x0f40, 0, 4, (char *)&clrdata, NULL, 1, NULL);
lspmac_send_command( VR_UPLOAD, VR_PMAC_SETMEM
    , 0x0e9c, 0, 4, (char *)&clrdata, NULL, 1, NULL);

lspmac_send_command( VR_UPLOAD, VR_PMAC_SETMEM
    , 0x0ea0, 0, strlen(qp->p1)+1, qp->p1, NULL, 1, NULL);

mask = 0x0001;
lspmac_send_command( VR_UPLOAD, VR_PMAC_SETBIT
    , 0x0e9c, 1, sizeof( mask), (char *)&mask,lspmac_asciicmdcb, 1,
    qp->event);

if( qp->event != NULL && *(qp->event) != 0)
    lsevents_send_event( "%s queued", qp->event);
}

```

#### 7.5.4.89 pmac\_cmd\_queue\_t\* lspmac\_SockSendline( char \* event, char \* fmt, ... )

Send a one line command.

Uses printf style arguments.

##### Parameters

in	<i>event</i>	base name for events
in	<i>fmt</i>	Printf style format string

Definition at line 1092 of file lspmac.c.

```

    {
va_list arg_ptr;
char payload[1400];

```

```

va_start( arg_ptr, fmt);
vsnprintf( payload, sizeof(payload)-1, fmt, arg_ptr);
payload[ sizeof(payload)-1 ] = 0;
va_end( arg_ptr);

lslogging_log_message( "%s", payload);

return lspmac_send_command( VR_DOWNLOAD,
    VR_PMAC_SENDLINE, 0, 0, strlen( payload), payload,
    lspmac_GetShortReplyCB, 0, event);
}

```

#### 7.5.4.90 pmac\_cmd\_queue\_t\* lspmac\_SockSendline\_nr ( char \* event, char \* fmt, ... )

Send a command and ignore the response.

##### Parameters

in	<i>event</i>	base name for events
in	<i>fmt</i>	Printf style format string

Definition at line 1115 of file lspmac.c.

```

{
va_list arg_ptr;
char s[512];

va_start( arg_ptr, fmt);
vsnprintf( s, sizeof(s)-1, fmt, arg_ptr);
s[sizeof(s)-1] = 0;
va_end( arg_ptr);

lslogging_log_message( s);

return lspmac_send_command( VR_DOWNLOAD,
    VR_PMAC_SENDLINE, 0, 0, strlen( s), s, NULL, 1, event);
}

```

#### 7.5.4.91 lspmac\_motor\_t\* lspmac\_soft\_motor\_init ( lspmac\_motor\_t \* d, char \* name, int(\*) (lspmac\_motor\_t \*, double) moveAbs )

Definition at line 3745 of file lspmac.c.

```

{
lspmac_motor_init( d, name);

d->moveAbs      = moveAbs;
d->jogAbs       = moveAbs;
d->read         = lspmac_soft_motor_read;
d->actual_pos_ctns_p = calloc( sizeof(int), 1);
*d->actual_pos_ctns_p = 0;

return d;
}

```

#### 7.5.4.92 void lspmac\_soft\_motor\_read ( lspmac\_motor\_t \* p )

Dummy routine to read a soft motor.

Definition at line 3740 of file lspmac.c.

```

{
}
```

#### 7.5.4.93 int lsmpmac\_test\_preset ( *lsmpmac\_motor\_t* \* *mp*, char \* *preset\_name*, double *tolerance* )

see if the motor is within tolerance of the preset 1 means yes, it is 0 mean no it isn't or that the preset was not found  
 Definition at line 2496 of file lsmpmac.c.

```
{
double preset_position;
int err;

err = lsredis_find_preset( mp->name, preset_name, &
    preset_position);
if( err == 0)
    return 0;

if( fabs( preset_position - lsmpmac_getPosition( mp)) <=
    tolerance)
    return 1;

return 0;
}
```

#### 7.5.4.94 void lsmpmac\_video\_rotate ( double *secs* )

Special motion program to collect centering video.

Definition at line 2705 of file lsmpmac.c.

```
{
double q10;           // starting position (counts)
double q11;           // delta counts
double q12;           // milliseconds to run over delta

double u2c;
double neutral_pos;

if( secs <= 0.0)
    return;

omega_zero_search = 1;

pthread_mutex_lock( &(omega->mutex));
u2c      = lsredis_getd( omega->u2c);
neutral_pos = lsredis_getd( omega->neutral_pos);

q10 = neutral_pos * u2c;
q11 = 360.0 * u2c;
q12 = 1000 * secs;

omega_zero_velocity = 360.0 * u2c / secs; // 
counts/second to back calculate zero crossing time

lsmpmac_SockSendDLine( omega->name, "&
    Q10=%1f Q11=%1f Q12=%1f Q13=(I117) Q14=(I116) B240R", q10, q11, q12);
pthread_mutex_unlock( &(omega->mutex));
}
```

#### 7.5.4.95 void\* lsmpmac\_worker ( void \* *dummy* )

Our lsmpmac worker thread.

##### Parameters

in	<i>dummy</i>	Unused but required by pthread library
----	--------------	--

Definition at line 2331 of file lsmpmac.c.

```
{
static int disconnected_notify = 0;
static int old_state;
```

```

old_state = ls_pmac_state;
while( lspmac_running) {
    int pollrtn;

    lspmac_next_state();

    if( ls_pmac_state != old_state) {
        // lslogging_log_message( "lspmac_worker: state = %d",
        // ls_pmac_state);
        old_state = ls_pmac_state;
    }

    if( pmacfd.fd == -1) {
        if( disconnected_notify == 0)
            lslogging_log_message( "lspmac_worker: PMAC not
connected");
        disconnected_notify = 1;
        //
        // At this point we assume we became disconnected due to something like a
        hard boot of the MD2 PMAC
        // and hence the entire system needs reinitialization.
        //
        // It's possible to put in a test here (perhaps using I65) to see if we
        in fact suffered a reset
        // and need to clear the queue, reinitialize, etc. Or if it was just a
        networking glitch and do not
        // need to clear the queue and should instead just charge ahead.
        //
        lspmac_reset_queue();
        sleep( 10);
        //
        // This just puts us into a holding pattern until the pmac becomes
        connected again
        //
        continue;
    }
    disconnected_notify = 0;

    pollrtn = poll( &pmacfd, 1, 10);
    if( pollrtn) {
        lspmac_Service( &pmacfd);
    }
}
pthread_exit( NULL);
}

```

#### 7.5.4.96 void lspmac\_zoom\_lut\_setup( )

Set up lookup table for zoom.

Definition at line 4184 of file lspmac.c.

```

{
int i;
lsredis_obj_t *p;
double neutral_pos;

neutral_pos = lsredis_getd( zoom->neutral_pos);

pthread_mutex_lock( &zoom->mutex);

zoom->nlut = 10;
zoom->lut = calloc( 2 * zoom->nlut, sizeof( double));
if( zoom->lut == NULL) {
    lslogging_log_message( "lspmac_zoom_lut_setup: out of
memory");
    exit( -1);
}

for( i=0; i < zoom->nlut; i++) {
    p = lsredis_get_obj( "cam.zoom.%d.MotorPosition", i+1);
    if( p==NULL || strlen( lsredis_getstr(p)) == 0) {
        free( zoom->lut);
        zoom->lut = NULL;
        zoom->nlut = 0;
        pthread_mutex_unlock( &zoom->mutex);
        lslogging_log_message( "lspmac_zoom_lut_setup:
cannot find MotorPosition element for cam.zoom level %d", i+1);
        return;
    }
    zoom->lut[2*i] = i+1;
}

```

```

    zoom->lut[2*i+1] = lsredis_getd( p ) + neutral_pos;
}
pthread_mutex_unlock( &zoom->mutex );
}

```

## 7.5.5 Variable Documentation

### 7.5.5.1 **lspmac\_motor\_t\* alignx**

Alignment stage X.

Definition at line 103 of file lspmac.c.

### 7.5.5.2 **lspmac\_motor\_t\* aligny**

Alignment stage Y.

Definition at line 104 of file lspmac.c.

### 7.5.5.3 **lspmac\_motor\_t\* alignz**

Alignment stage X.

Definition at line 105 of file lspmac.c.

### 7.5.5.4 **lspmac\_motor\_t\* anal**

Polaroid analyzer motor.

Definition at line 106 of file lspmac.c.

### 7.5.5.5 **lspmac\_motor\_t\* apery**

Aperture Y.

Definition at line 108 of file lspmac.c.

### 7.5.5.6 **lspmac\_motor\_t\* aperz**

Aperture Z.

Definition at line 109 of file lspmac.c.

### 7.5.5.7 **lspmac\_bi\_t\* arm\_parked**

(whose arm? parked where?)

Definition at line 146 of file lspmac.c.

### 7.5.5.8 **lspmac\_motor\_t\* blight**

Back Light DAC.

Definition at line 120 of file lspmac.c.

**7.5.5.9 Ispmac.bi\_t\* blight\_down**

Backlight is down.

Definition at line 136 of file Ispmac.c.

**7.5.5.10 Ispmac\_motor\_t\* blight\_f**

Back light scale factor.

Definition at line 129 of file Ispmac.c.

**7.5.5.11 Ispmac\_motor\_t\* blight\_ud**

Back light Up/Down actuator.

Definition at line 124 of file Ispmac.c.

**7.5.5.12 Ispmac.bi\_t\* blight\_up**

Backlight is up.

Definition at line 137 of file Ispmac.c.

**7.5.5.13 Ispmac\_motor\_t\* capy**

Capillary Y.

Definition at line 110 of file Ispmac.c.

**7.5.5.14 Ispmac\_motor\_t\* capz**

Capillary Z.

Definition at line 111 of file Ispmac.c.

**7.5.5.15 Ispmac\_motor\_t\* cenx**

Centering Table X.

Definition at line 113 of file Ispmac.c.

**7.5.5.16 Ispmac\_motor\_t\* ceny**

Centering Table Y.

Definition at line 114 of file Ispmac.c.

**7.5.5.17 pmac\_cmd\_t cr\_cmd [static]**

commands to send out "readready", "getbuffer", "controlresponse" (initialized in main)

Definition at line 198 of file Ispmac.c.

---

**7.5.5.18 `Ispmac_motor_t* cryo`**

Move the cryostream towards or away from the crystal.

Definition at line 125 of file Ispmac.c.

**7.5.5.19 `Ispmac_bi_t* cryo_back`**

cryo is in the back position

Definition at line 138 of file Ispmac.c.

**7.5.5.20 `Ispmac_bi_t* cryo_switch`**

that little toggle switch for the cryo

Definition at line 135 of file Ispmac.c.

**7.5.5.21 `unsigned char dbmem[64 *1024] [static]`**

double buffered memory

Definition at line 187 of file Ispmac.c.

**7.5.5.22 `int dbmemln = 0 [static]`**

next location

Definition at line 188 of file Ispmac.c.

**7.5.5.23 `Ispmac_motor_t* dryer`**

blow air on the scintillator to dry it off

Definition at line 126 of file Ispmac.c.

**7.5.5.24 `Ispmac_bi_t* etel_init_ok`**

ETEL initialized OK.

Definition at line 143 of file Ispmac.c.

**7.5.5.25 `Ispmac_bi_t* etel_on`**

ETEL is on.

Definition at line 142 of file Ispmac.c.

**7.5.5.26 `Ispmac_bi_t* etel_ready`**

ETEL is ready.

Definition at line 141 of file Ispmac.c.

7.5.5.27 `unsigned int ethCmdOff = 0 [static]`

points to current command (or none if == ethCmdOn)

Definition at line 201 of file lspmac.c.

7.5.5.28 `unsigned int ethCmdOn = 0 [static]`

points to next empty PMAC command queue position

Definition at line 200 of file lspmac.c.

7.5.5.29 `pmac_cmd_queue_t ethCmdQueue[PMAC_CMD_QUEUE_LENGTH] [static]`

PMAC command queue.

Definition at line 199 of file lspmac.c.

7.5.5.30 `unsigned int ethCmdReply = 0 [static]`

Used like ethCmdOff only to deal with the pmac reply to a command.

Definition at line 202 of file lspmac.c.

7.5.5.31 `lspmac_motor_t* flight`

Front Light DAC.

Definition at line 119 of file lspmac.c.

7.5.5.32 `lspmac_motor_t* flight_f`

Front light scale factor.

Definition at line 130 of file lspmac.c.

7.5.5.33 `lspmac_motor_t* flight_oo`

Turn front light on/off.

Definition at line 128 of file lspmac.c.

7.5.5.34 `lspmac_motor_t* fluo`

Move the fluorescence detector in/out.

Definition at line 127 of file lspmac.c.

7.5.5.35 `lspmac_bi_t* fluor_back`

fluor is in the back position

Definition at line 139 of file lspmac.c.

**7.5.5.36 `Ispmac_motor_t* fscint`**

Scintillator Piezo DAC.

Definition at line 121 of file `Ispmac.c`.

**7.5.5.37 `Ispmac_motor_t* fshut`**

Fast shutter.

Definition at line 118 of file `Ispmac.c`.

**7.5.5.38 `pmac_cmd_t gb_cmd [static]`**

Definition at line 198 of file `Ispmac.c`.

**7.5.5.39 `int getivars = 0 [static]`**

flag set at initialization to send i vars to db

Definition at line 91 of file `Ispmac.c`.

**7.5.5.40 `int getmvars = 0 [static]`**

flag set at initialization to send m vars to db

Definition at line 92 of file `Ispmac.c`.

**7.5.5.41 `Ispmac_bi_t* hp_air`**

High pressure air OK.

Definition at line 134 of file `Ispmac.c`.

**7.5.5.42 `Ispmac_motor_t* kappa`**

Kappa.

Definition at line 115 of file `Ispmac.c`.

**7.5.5.43 `Ispmac_bi_t* lp_air`**

Low pressure air OK.

Definition at line 133 of file `Ispmac.c`.

**7.5.5.44 `int ls_pmac_state = LS_PMAC_STATE_DETACHED [static]`**

Current state of the PMAC communications state machine.

Definition at line 58 of file `Ispmac.c`.

**7.5.5.45 `Ispmac_ascii_buffers_t Ispmac_ascii_buffers [static]`**

Definition at line 367 of file `Ispmac.c`.

7.5.5.46 `pthread_mutex_t lspmac_ascii_buffers_mutex`

Definition at line 368 of file lspmac.c.

7.5.5.47 `int lspmac_ascii_busy = 0 [static]`

flag for condition to wait for

Definition at line 78 of file lspmac.c.

7.5.5.48 `pthread_mutex_t lspmac_ascii_mutex [static]`

Keep too many processes from sending commands at once.

Definition at line 77 of file lspmac.c.

7.5.5.49 `lspmac_bi_t lspmac_bis[32]`

array of binary inputs

Definition at line 94 of file lspmac.c.

7.5.5.50 `uint16_t lspmac_control_char = 0 [static]`

The control character we've sent.

Definition at line 75 of file lspmac.c.

7.5.5.51 `uint32_t lspmac_dpascii_off = 0 [static]`

Definition at line 378 of file lspmac.c.

7.5.5.52 `uint32_t lspmac_dpascii_on = 0 [static]`

Definition at line 377 of file lspmac.c.

7.5.5.53 `lspmac_dpascii_queue_t lspmac_dpascii_queue[LSPMAC_DPASCII_QUEUE_LENGTH] [static]`

Definition at line 376 of file lspmac.c.

7.5.5.54 `lspmac_motor_t lspmac_motors[LSPMAC_MAX_MOTORS]`

All our motors.

Definition at line 98 of file lspmac.c.

7.5.5.55 `pthread_cond_t lspmac_moving_cond`

Wait for motor(s) to finish moving condition.

Definition at line 72 of file lspmac.c.

**7.5.5.56 int lspmac\_moving\_flags**

Flag used to implement motor moving condition.

Definition at line 73 of file lspmac.c.

**7.5.5.57 pthread\_mutex\_t lspmac\_moving\_mutex**

Coordinate moving motors between threads.

Definition at line 71 of file lspmac.c.

**7.5.5.58 int lspmac\_nbis = 0**

number of active binary inputs

Definition at line 95 of file lspmac.c.

**7.5.5.59 int lspmac\_nmotors = 0**

The number of motors we manage.

Definition at line 99 of file lspmac.c.

**7.5.5.60 int lspmac\_running = 1 [static]**

exit worker thread when zero

Definition at line 66 of file lspmac.c.

**7.5.5.61 pthread\_cond\_t lspmac\_shutter\_cond**

Allows waiting for the shutter status to change.

Definition at line 70 of file lspmac.c.

**7.5.5.62 int lspmac\_shutter\_has\_opened**

Indicates that the shutter had opened, perhaps briefly even if the state did not change.

Definition at line 68 of file lspmac.c.

**7.5.5.63 pthread\_mutex\_t lspmac\_shutter\_mutex**

Coordinates threads reading shutter status.

Definition at line 69 of file lspmac.c.

**7.5.5.64 int lspmac\_shutter\_state**

State of the shutter, used to detect changes.

Definition at line 67 of file lspmac.c.

7.5.5.65 `struct timespec lspmac_status.last_time [static]`

Time the status was read.

Definition at line 84 of file lspmac.c.

7.5.5.66 `struct timespec lspmac_status_time [static]`

Time the status was read.

Definition at line 83 of file lspmac.c.

7.5.5.67 `md2_status_t md2_status [static]`

Buffer for MD2 Status.

Definition at line 353 of file lspmac.c.

7.5.5.68 `pthread_mutex_t md2_status_mutex`

Synchronize reading/writing status buffer.

Definition at line 354 of file lspmac.c.

7.5.5.69 `lspmac_bi_t* minikappa_ok`

Minikappa is OK (whatever that means)

Definition at line 144 of file lspmac.c.

7.5.5.70 `struct hsearch_data motors_ht`

A hash table to find motors by name.

Definition at line 100 of file lspmac.c.

7.5.5.71 `struct timeval pmac_time_sent now [static]`

used to ensure we do not send commands to the pmac too often. Only needed for non-DB commands.

Definition at line 194 of file lspmac.c.

7.5.5.72 `lspmac_motor_t* omega`

MD2 omega axis (the air bearing)

Definition at line 102 of file lspmac.c.

7.5.5.73 `int omega_zero_search = 0 [static]`

Indicate we'd really like to know when omega crosses zero.

Definition at line 80 of file lspmac.c.

#### 7.5.5.74 struct timespec omega\_zero\_time

Time we believe that omega crossed zero.

Definition at line 82 of file lspmacc.c.

#### 7.5.5.75 double omega\_zero\_velocity = 0 [static]

rate (cnts/sec) that omega was traveling when it crossed zero

Definition at line 81 of file lspmacc.c.

#### 7.5.5.76 lspmacc\_motor\_t\* phi

Phi (not data collection axis)

Definition at line 116 of file lspmacc.c.

#### 7.5.5.77 char\* pmac\_error\_strs[] [static]

##### Initial value:

```
= {
    "ERR000: Unknown error",
    "ERR001: Command not allowed during program execution",
    "ERR002: Password error",
    "ERR003: Data error or unrecognized command",
    "ERR004: Illegal character",
    "ERR005: Command not allowed unless buffer is open",
    "ERR006: No room in buffer for command",
    "ERR007: Buffer already in use",
    "ERR008: MACRO auxiliary communication error",
    "ERR009: Program structure error (e.g. ENDIF without IF)",
    "ERR010: Both overtravel limits set for a motor in the C.S.",
    "ERR011: Previous move not completed",
    "ERR012: A motor in the coordinate system is open-loop",
    "ERR013: A motor in the coordinate system is not activated",
    "ERR014: No motors in the coordinate system",
    "ERR015: Not pointer to valid program buffer",
    "ERR016: Running improperly structure program (e.g. missing ENDWHILE)",
    "ERR017: Trying to resume after H or Q with motors out of stopped position",
    "ERR018: Attempt to perform phase reference during move, move during phase
            reference, or enabling with phase clock error",
    "ERR019: Illegal position-change command while moves stored in CCBUFFER",
    "ERR020: FSAVE issued on Turbo PMAC with incompatible flash memory",
    "ERR021: FSAVE issued while clearing old flash memory sector",
    "ERR022: FREAD attempted but the flash memory is bad"
}
```

Decode the errors perhaps returned by the PMAC.

Definition at line 205 of file lspmacc.c.

#### 7.5.5.78 pthread\_cond\_t pmac\_queue\_cond

wait for a command to be sent to PMAC before continuing

Definition at line 88 of file lspmacc.c.

#### 7.5.5.79 pthread\_mutex\_t pmac\_queue\_mutex

manage access to the pmac command queue

Definition at line 87 of file lspmacc.c.

**7.5.5.80 pthread\_t pmac\_thread [static]**

our thread to manage access and communication to the pmac

Definition at line 86 of file lspmac.c.

**7.5.5.81 struct pollfd pmacfd [static]**

our poll structure

Definition at line 89 of file lspmac.c.

**7.5.5.82 pmac\_cmd\_t rr\_cmd [static]**

Definition at line 198 of file lspmac.c.

**7.5.5.83 lspmac\_bi\_t\* sample\_detected**

smart magnet detected sample

Definition at line 140 of file lspmac.c.

**7.5.5.84 lspmac\_motor\_t\* scint**

Scintillator Z.

Definition at line 112 of file lspmac.c.

**7.5.5.85 lspmac\_bi\_t\* shutter\_open**

shutter is open (note in pmc says this is a slow input)

Definition at line 147 of file lspmac.c.

**7.5.5.86 lspmac\_bi\_t\* smart\_mag\_err**

smart magnet error (coil broken perhaps)

Definition at line 148 of file lspmac.c.

**7.5.5.87 lspmac\_bi\_t\* smart\_mag\_off**

smart magnet is off

Definition at line 149 of file lspmac.c.

**7.5.5.88 lspmac\_bi\_t\* smart\_mag\_on**

smart magnet is on

Definition at line 145 of file lspmac.c.

### 7.5.5.89 `lspmac_motor_t* smart_mag_oo`

Smart Magnet on/off.

Definition at line 123 of file `lspmac.c`.

### 7.5.5.90 `lspmac_motor_t* zoom`

Optical zoom.

Definition at line 107 of file `lspmac.c`.

## 7.6 `lsredis.c` File Reference

Support redis hash synchronization.

```
#include "pgpmac.h"
```

### Data Structures

- struct `lsredis_preset_list_struct`

### Typedefs

- typedef struct  
`lsredis_preset_list_struct lsredis_preset_list_t`

### Functions

- void `lsredis_debugCB` (redisAsyncContext \*ac, void \*reply, void \*privdata)  
*Log the reply.*
- void `_lsredis_set_value` (`lsredis_obj_t` \*p, char \*v)  
*set\_value and setstr helper funciton p->mutex must be locked before calling*
- void `lsredis_set_value` (`lsredis_obj_t` \*p, char \*fmt,...)  
*Set the value of a redis object and make it valid.*
- int `lsredis_cmpstr` (`lsredis_obj_t` \*p, char \*s)
- int `lsredis_cmpnstr` (`lsredis_obj_t` \*p, char \*s, int n)
- int `lsredis_reexec` (const regex\_t \*preg, `lsredis_obj_t` \*p, size\_t nmatch, regmatch\_t \*pmatch, int eflags)  
*return a copy of the key's string value be sure to free the result*
- char \* `lsredis_getstr` (`lsredis_obj_t` \*p)  
*Set the value and update redis.*
- double `lsredis_get_or_set_d` (`lsredis_obj_t` \*p, double val, int prec)
- double `lsredis_getd` (`lsredis_obj_t` \*p)
- long int `lsredis_getl` (`lsredis_obj_t` \*p)
- long int `lsredis_get_or_set_l` (`lsredis_obj_t` \*p, long int val)
- char \*\* `lsredis_get_string_array` (`lsredis_obj_t` \*p)
- int `lsredis_getb` (`lsredis_obj_t` \*p)
- char `lsredis_getc` (`lsredis_obj_t` \*p)
- void `lsredis_hgetCB` (redisAsyncContext \*ac, void \*reply, void \*privdata)
- `lsredis_obj_t * _lsredis_get_obj` (char \*key)

- Maybe add a new object Used internally for this module Must be called with lsredis\_mutex locked.*
- `lsredis_obj_t * lsredis_get_obj (char *fmt,...)`
  - `void redisDisconnectCB (const redisAsyncContext *ac, int status)`

*call back in case a redis server becomes disconnected TODO: reconnect*
  - `void lsredis_addRead (void *data)`

*hook to mange read events*
  - `void lsredis_delRead (void *data)`

*hook to manage "don't need to read" events*
  - `void lsredis_addWrite (void *data)`

*hook to manage write events*
  - `void lsredis_delWrite (void *data)`

*hook to manage "don't need to write anymore" events*
  - `void lsredis_cleanup (void *data)`

*hook to clean up TODO: figure out what we are supposed to do here and do it*
  - `void lsredis_subCB (redisAsyncContext *ac, void *reply, void *privdata)`

*Use the publication to request the new value.*
  - `void lsredis_maybe_add_key (char *k)`
  - `void lsredis_keysCB (redisAsyncContext *ac, void *reply, void *privdata)`

*Sift through the keys to find ones we like.*
  - `void lsredis_load_presets (char *motor_name)`

*update the presets hash table for the named motor*
  - `int lsredis_find_preset (char *motor_name, char *preset_name, double *dval)`

*Get the value of the given preset and return it in dval Returns 0 on error, non-zero on success.:*
  - `void lsredis_set_preset (char *motor_name, char *preset_name, double dval)`

*set the given preset to the given value create a new preset if we can't find it*
  - `int lsredis_find_preset_index_by_position (lspmac_motor_t *mp)`

*For the given motor object return the index of the current preset or -1 if we are not at a preset position.*
  - `void lsredis_configCB (redisAsyncContext *ac, void *reply, void *privdata)`
  - `void lsredis_config ()`
  - `void lsredis_init ()`

*Initialize this module, that is, set up the connections.*
  - `void lsredis_fd_service (struct pollfd *evt)`

*service the socket requests*
  - `void lsredis_sig_service (struct pollfd *evt)`
  - `void * lsredis_worker (void *dummy)`

*subscribe to changes and service sockets*
  - `void lsredis_run ()`

## Variables

- `static pthread_t lsredis_thread`
- `pthread_mutex_t lsredis_mutex = PTHREAD_RECURSIVE_MUTEX_INITIALIZER_NP`
- `pthread_cond_t lsredis_cond`
- `int lsredis_running = 0`
- `static pthread_mutexattr_t mutex_initializer`
- `static lsredis_obj_t * lsredis_objs = NULL`
- `static struct hsearch_data lsredis_htab`
- `static redisAsyncContext * subac`
- `static redisAsyncContext * roac`
- `static redisAsyncContext * wrac`
- `static char * lsredis_publisher = NULL`
- `static regex_t lsredis_key_select_regex`

- static char \* `lsredis_head` = NULL
- static pthread\_mutex\_t `lsredis_config_mutex`
- static pthread\_cond\_t `lsredis_config_cond`
- static struct pollfd `subfd`
- static struct pollfd `rnofd`
- static struct pollfd `wrfd`
- static `lsredis_preset_list_t` \* `lsredis_preset_list` = NULL
- static struct hsearch\_data `lsredis_preset_ht`
- static int `lsredis_preset_n` = 0
- static int `lsredis_preset_max_n` = 1024
- static pthread\_mutex\_t `lsredis_preset_list_mutex`

### 7.6.1 Detailed Description

Support redis hash synchronization.

```
\date 2012
\author Keith Brister
\copyright All Rights Reserved
```

Redis support for redis in pgpmac.

Values in redis are assumed to be hashes with at least one field "VALUE". At startup the initialization routine is passed a regular expression to select which keys we'd like to duplicate locally as a `lsredis_obj_t`. It is assumed that the following construct in redis is used to change a value:

```
MULTI
HSET key VALUE value
PUBLISH publisher key
EXEC
```

Where "publisher" is a unique name in the following format:

```
MD2-*
or    UI-*
or    REDIS_KV_CONNECTOR
or    mk_pgpmac_redis
```

(this last value is used to support the now deprecated px.kvs table in the LS-CAT postgresql server). We assume that all publishers that we are listening to ONLY publish key names that have changed.

When someone else changes a value we invalidate our internal copy and issue a "HGET key VALUE" command. Other threads that request the value of our `lsredis_obj_t` will pause until the new value has been received and processed.

When a value changes locally this module changes it in redis as shown above. At this point we refuse other publishers attempt to change the value until we've seen all of our PUBLISH messages. That is, we ignore changes that in redis happened before our change.

You'll need an `lsredis_obj_t` to do anything with redis in the pgpmac project:

```
lsredis_obj_t *lsredis_get_obj( char *fmt, ...) where fmt is a printf style formatting string  

During initialization a "head" string is passed  

For example, "omega.position" might refer to t
```

To set a redis value use

```
void lsredis_setstr( lsredis_obj_t *p, char *fmt, ...) where fmt is a printf style formatting
```

When a new value is seen we immediately parse it and make it available through the following functions:

char *lsredis_getstr( lsredis_obj_t *p)	Returns a copy of the VALUE field. Used for strings.
double lsredis_getd( lsredis_obj_t *p)	Returns a double. If the value was not a valid double, returns 0.0.
long int lsredis_getl( lsredis_obj_t *p)	Returns a long int. If the value was not a valid long int, returns 0.
char **lsredis_get_string_array( lsredis_obj_t *p)	Returns an array of string pointers. Returns NULL if the value could not be parsed.
int lsredis_getb( lsredis_obj_t *p)	Returns 1, 0, or -1 based on the first character of VALUE.
char lsredis_getc( lsredis_obj_t *p)	Returns the first character of VALUE.

Definition in file [lsredis.c](#).

## 7.6.2 Typedef Documentation

### 7.6.2.1 `typedef struct lsredis_preset_list_struct lsredis_preset_list_t`

## 7.6.3 Function Documentation

### 7.6.3.1 `lsredis_obj_t* _lsredis_get_obj( char * key )`

Maybe add a new object Used internally for this module Must be called with lsredis\_mutex locked.

Definition at line 510 of file [lsredis.c](#).

```

{
lsredis_obj_t *p;
regmatch_t pmatch[2];
int err;
ENTRY htab_input, *htab_output;

// Dispense with obviously bad keys straight away
// unless p->valid == 0 in which case we call HGET first
//
// TODO: review logic: is there ever a time when valid is zero for a
// preexisting p and HGET has not been called?
//     If not then we should just return p without checking for validity.
//
if( key == NULL || *key == 0 || strchr( key, ' ' ) != NULL) {
    lslogging_log_message( "_lsredis_get_obj: bad key '%s'
    ", key == NULL ? "<NULL>" : key);
    return NULL;
}

// If the key is already there then just return it
//

htab_input.key = key;
htab_input.data = NULL;
errno = 0;
err = hsearch_r( htab_input, FIND, &htab_output, &lsredis_htab);

if( err == 0)
    p = NULL;
else
    p = htab_output->data;

if( p != NULL)
    return p;
}

```

```

} else {
    // make a new one.
    p = calloc( 1, sizeof( lsredis_obj_t));
    if( p == NULL) {
        lslogging_log_message( "_lsredis_get_obj: Out of
            memory");
        exit( -1);
    }

    err = regexec( &lsredis_key_select_regex, key, 2,
        pmatch, 0);
    if( err == 0 && pmatch[1].rm_so != -1) {
        p->events_name = strdup( key+pmatch[1].rm_so, pmatch[1].rm_eo
            - pmatch[1].rm_so);
    } else {
        p->events_name = strdup( key);
    }
    if( p->events_name == NULL) {
        lslogging_log_message( "_lsredis_get_obj: Out of
            memory (events_name)");
        exit( -1);
    }

    pthread_mutex_init( &p->mutex, &mutex_initializer);
    pthread_cond_init( &p->cond, NULL);
    p->value = NULL;
    p->valid = 0;
    lsevents_send_event( "%s Invalid", p->events_name
        );
    p->wait_for_me = 0;
    p->key = strdup( key);
    p->hits = 0;

    htab_input.key = p->key;
    htab_input.data = p;

    errno = 0;
    err = hsearch_r( htab_input, ENTER, &htab_output, &lsredis_htab
        );
    if( err == 0) {
        lslogging_log_message( "_lsredis_get_obj: hsearch
            error on enter.  errno=%d", errno);
    }
}

// Shouldn't need the linked list unless we need to rebuild the hash table
// when, for example, we run out of room.
// TODO: resize hash table when needed.
//
p->next = lsredis_objs;
lsredis_objs = p;
}
//
// We arrive here with the valid flag lowered.  Go ahead and request the
// latest value.
//
redisAsyncCommand( roac, lsredis_hgetCB, p, "HGET %s VALUE"
    , key);

return p;
}

```

### 7.6.3.2 void \_lsredis\_set\_value( lsredis\_obj\_t \*p, char \*v )

set\_value and setstr helper funciton p->mutex must be locked before calling

Definition at line 166 of file lsredis.c.

```

{
if( strlen(v) >= (unsigned int) p->value_length) {
    if( p->value != NULL)
        free( p->value);
    p->value_length = strlen(v) + 256;
    p->value = calloc( p->value_length, sizeof( char));
    if( p->value == NULL) {
        lslogging_log_message( "_lsredis_set_value: out of
            memory");
        exit( -1);
    }
}
strncpy( p->value, v, p->value_length - 1);

```

```

p->value[p->value_length-1] = 0;
p->dvalue = strtod( p->value, NULL);
p->lvalue = p->dvalue;

if( p->avalue != NULL) {
    int i;
    for( i=0; (p->avalue)[i] != NULL; i++)
        free( (p->avalue)[i]);
    free( p->avalue);
    p->avalue = NULL;
}

p->avalue = lsgc_array2ptrs( p->value);

switch( *(p->value)) {
    case 'T':
    case 't':
    case 'Y':
    case 'y':
    case '1':
        p->bvalue = 1;
        break;

    case 'F':
    case 'f':
    case 'N':
    case 'n':
    case '0':
        p->bvalue = 0;
        break;

    default:
        p->bvalue = -1;           // nil is -1 here in our world
}
}

p->cvalue = *(p->value);

if( !(p->valid)) {
    p->valid = 1;
    lsevents_send_event( "%s Valid", p->events_name
)
}
}
}

```

### 7.6.3.3 void lsredis\_addRead( void \* data )

hook to mange read events

Definition at line 640 of file lsredis.c.

```

{
struct pollfd *pfd;
pfd = (struct pollfd *)data;

if( (pfd->events & POLLIN) == 0) {
    pfd->events |= POLLIN;
    pthread_kill( lsredis_thread, SIGUSR1);
}
}

```

### 7.6.3.4 void lsredis\_addWrite( void \* data )

hook to manage write events

Definition at line 664 of file lsredis.c.

```

{
struct pollfd *pfd;
pfd = (struct pollfd *)data;

if( (pfd->events & POLLOUT) == 0) {
    pfd->events |= POLLOUT;
    pthread_kill( lsredis_thread, SIGUSR1);
}
}

```

### 7.6.3.5 void lsredis\_cleanup ( void \* data )

hook to clean up TODO: figure out what we are supposed to do here and do it

Definition at line 689 of file lsredis.c.

```

    {
    struct pollfd *pfd;
    pfd = (struct pollfd *)data;

    pfd->fd = -1;

    if( (pfd->events & (POLLOUT | POLLIN)) != 0 ) {
        pfd->events &= ~(POLLOUT | POLLIN);
        pthread_kill( lsredis_thread, SIGUSR1 );
    }
}

```

### 7.6.3.6 int lsredis\_cmpnstr ( lsredis\_obj\_t \* p, char \* s, int n )

Definition at line 256 of file lsredis.c.

```

    {
int rtn;
pthread_mutex_lock( &p->mutex );
while( p->valid == 0 )
    pthread_cond_wait( &p->cond, &p->mutex );

rtn = strncmp( p->value, s, n );
pthread_mutex_unlock( &p->mutex );
return rtn;
}

```

### 7.6.3.7 int lsredis\_cmpstr ( lsredis\_obj\_t \* p, char \* s )

Definition at line 245 of file lsredis.c.

```

    {
int rtn;
pthread_mutex_lock( &p->mutex );
while( p->valid == 0 )
    pthread_cond_wait( &p->cond, &p->mutex );

rtn = strcmp( p->value, s );
pthread_mutex_unlock( &p->mutex );
return rtn;
}

```

### 7.6.3.8 void lsredis\_config ( )

Definition at line 1097 of file lsredis.c.

```

    {
char hostname[128], lhostname[128];
int i;

pthread_mutexattr_init( &mutex_initializer );
pthread_mutexattr_settype( &mutex_initializer,
    PTHREAD_MUTEX_RECURSIVE );

if( gethostname( hostname, sizeof(hostname)-1 ) ) {
    lslogging_log_message( "lsredis_init: cannot get our
        own host name. Cannot configure redis variables." );
} else {
    for( i=0; i<strlen(hostname); i++ ) {
        lhostname[i] = tolower( hostname[i] );
    }
    lhostname[i] = 0;
}

```

```

lslogging_log_message( "lsredis_init: our host name is
    '%s', lhostname);
redisAsyncCommand( roac, lsredis_configCB, NULL,
    hgetall config.%s", lhostname);
}

pthread_mutex_lock( &lsredis_config_mutex);
while( lsredis_head == NULL)
    pthread_cond_wait( &lsredis_config_cond, &
        lsredis_config_mutex);
pthread_mutex_unlock( &lsredis_config_mutex);

}

```

### 7.6.3.9 void lsredis\_configCB ( redisAsyncContext \* ac, void \* reply, void \* privdata )

Definition at line 1012 of file lsredis.c.

```

{
redisReply *r, *r2, *r3;
int i;
char *errmsg;
int err, nerrmsg;

r = reply;

if( r == NULL) {
    lslogging_log_message( "lsredis_configCB: null reply,
        cannot configure, bad things will happen");
    return;
}

if( r->type != REDIS_REPLY_ARRAY || (r->elements % 2) != 0) {
    lslogging_log_message( "lsredis_configCB: could not
        understand config reply. Bad things will happen.");
    return;
}

pthread_mutex_lock( &lsredis_config_mutex);

for( i=0; i<r->elements; i += 2) {
    r2 = r->element[i];
    r3 = r->element[i+1];
    if( r2->type != REDIS_REPLY_STRING || r2->type != REDIS_REPLY_STRING)
        continue;
    //
    // the LHS of the redis keys for this station
    //
    if( strcmp( r2->str, "HEAD")==0) {
        lsredis_head = strdup( r3->str);
    }
    //
    // Publish changes to keys with this name as the publisher
    //
    if( strcmp( r2->str, "PUB")==0) {
        lsredis_publisher = strdup( r3->str);
    }

    if( strcmp( r2->str, "PG")==0) {
        pgpmac_use_pg = r3->str[0] == '0' ? 0 : 1;
    }

    if( strcmp( r2->str, "AUTOSCINT")==0) {
        pgpmac_use_autoscint = r3->str[0] == '0' ? 0 : 1;
    }

    //
    // reg expression to select keys we will be keeping a local copy of
    //
    if( strcmp( r2->str, "RE")==0) {
        err = regcomp( &lsredis_key_select_regex, r3->str
        , REG_EXTENDED);
        if( err != 0) {
            nerrmsg = regerror( err, &lsredis_key_select_regex
            , NULL, 0);
            if( nerrmsg > 0) {
                errmsg = calloc( nerrmsg, sizeof( char));
                nerrmsg = regerror( err, &lsredis_key_select_regex
                , errmsg, nerrmsg);
                lslogging_log_message( "lsredis_configCB: %s",
                    errmsg);
                free( errmsg);
            }
        }
    }
}

```

```

        }
        exit( 1 );
    }
}

/*
2013-02-16 12:03:20.669342  ARRAY of 6 elements
2013-02-16 12:03:20.669351      STRING: HEAD
2013-02-16 12:03:20.669354      STRING: stns.2
2013-02-16 12:03:20.669355      STRING: RE
2013-02-16 12:03:20.669361      STRING: redis\.\kvseq\stns\.2\.(.+)
2013-02-16 12:03:20.669362      STRING: PUB
2013-02-16 12:03:20.669364      STRING: MD2-21-ID-E
*/

if( redisAsyncCommand( subac, lsredis_subCB, NULL, "
    PSUBSCRIBE REDIS_KV_CONNECTOR mk_pgpmac_redis UI* MD2-*") == REDIS_ERR) {
    lslogging_log_message( "Error sending PSUBSCRIBE
        command");
}
redisAsyncCommand( roac, lsredis_keysCB, NULL, "KEYS *");
pthread_cond_signal( &lsredis_config_cond);
pthread_mutex_unlock( &lsredis_config_mutex);
}

```

### 7.6.3.10 void lsredis\_debugCB ( redisAsyncContext \* ac, void \* reply, void \* privdata )

Log the reply.

Definition at line 116 of file lsredis.c.

```

{
static int indentlevel = 0;
redisReply *r;
int i;

r = (redisReply *)reply;

if( r == NULL) {
    lslogging_log_message( "Null reply. Odd");
    return;
}

switch( r->type) {
case REDIS_REPLY_STATUS:
    lslogging_log_message( "%*sSTATUS: %s", indentlevel*4,
        "", r->str);
    break;

case REDIS_REPLY_ERROR:
    lslogging_log_message( "%*sERROR: %s", indentlevel*4,
        "", r->str);
    break;

case REDIS_REPLY_INTEGER:
    lslogging_log_message( "%*sInteger: %lld", indentlevel
        *4, "", r->integer);
    break;

case REDIS_REPLY_NIL:
    lslogging_log_message( "%*s(nil)", indentlevel*4, "");
    break;

case REDIS_REPLY_STRING:
    lslogging_log_message( "%*sSTRING: %s", indentlevel*4,
        "", r->str);
    break;

case REDIS_REPLY_ARRAY:
    lslogging_log_message( "%*sARRAY of %d elements",
        indentlevel*4, "", (int)r->elements);
    indentlevel++;
    for( i=0; i<(int)r->elements; i++)
        lsredis_debugCB( ac, r->element[i], NULL);
    indentlevel--;
    break;

default:
    lslogging_log_message( "%*sUnknown type %d",

```

```
    indentlevel*4, "", r->type);  
}  
}
```

#### 7.6.3.11 void Isredis\_delRead ( void \* *data* )

hook to manage "don't need to read" events

Definition at line 652 of file lsredis.c.

```
    {  
        struct pollfd *pfds;  
        pfd = (struct pollfd *)data;  
  
        if( (pfds->events & POLLIN) != 0 ) {  
            pfd->events &= ~POLLIN;  
            pthread_kill( lsredis_thread, SIGUSR1 );  
        }  
    }  
}
```

#### 7.6.3.12 void lsredis\_delWrite ( void \* *data* )

hook to manage "don't need to write anymore" events

Definition at line 676 of file lsredis.c.

```
    struct pollfd *pfds;
    pfd = (struct pollfd *)data;

    if( (pfds->events & POLLOUT) != 0 ) {
        pfd->events &= ~POLLOUT;
        pthread_kill( lsredis_thread, SIGUSR1 );
    }
}
```

#### 7.6.3.13 void lsredis\_fd\_service ( struct pollfd \* evt )

service the socket requests

Definition at line 1199 of file lsredis.c.

```
pthread_mutex_lock( &lsredis_mutex );
if( evt->fd == subac->c.fd) {
    if( evt->revents & POLLIN)
        redisAsyncHandleRead( subac);
    if( evt->revents & POLLOUT)
        redisAsyncHandleWrite( subac);
}
if( evt->fd == roac->c.fd) {
    if( evt->revents & POLLIN)
        redisAsyncHandleRead( roac);
    if( evt->revents & POLLOUT)
        redisAsyncHandleWrite( roac);
}
if( evt->fd == wrac->c.fd) {
    if( evt->revents & POLLIN)
        redisAsyncHandleRead( wrac);
    if( evt->revents & POLLOUT)
        redisAsyncHandleWrite( wrac);
}
pthread_mutex_unlock( &lsredis_mutex );
}
```

#### 7.6.3.14 int lsredis\_find\_preset( char \* *motor\_name*, char \* *preset\_name*, double \* *dval* )

Get the value of the given preset and return it in *dval*. Returns 0 on error, non-zero on success.

Definition at line 903 of file lsredis.c.

```

{
char s[512];
int err;
ENTRY entry_in, *entry_outp;
lsredis_preset_list_t *pl;

snprintf( s, sizeof( s)-1, "%s%s", motor_name, preset_name );
s[sizeof(s)-1] = 0;

entry_in.key = s;
entry_in.data = NULL;
err = hsearch_r( entry_in, FIND, &entry_outp, &lsredis_preset_ht
);
if( err == 0 ) {
    // not found (or some other problem that means we don't have an answer
    //
    // Maybe someone added a new preset and we don't know about it yet
    //
    lsredis_load_presets( motor_name );
    err = hsearch_r( entry_in, FIND, &entry_outp, &lsredis_preset_ht
);
    if( err == 0 ) {
        //
        // Guess not. Give up. We tried
        //
        *dval = 0.0;
        return 0;
    }
}
pl = entry_outp->data;
*dval = lsredis_getd( pl->position );
return 1;
}

```

#### 7.6.3.15 int lsredis\_find\_preset\_index\_by\_position( lspmac\_motor\_t \* *mp* )

For the given motor object return the index of the current preset or -1 if we are not at a preset position.

Definition at line 985 of file lsredis.c.

```

{
lsredis_obj_t *p;
int plength;
int i;
double ur, pos;

p = lsredis_get_obj( "%s.presets.length", mp->name );
plength = lsredis_get_or_set_l( p, 0 );

if( plength <= 0 ) {
    return -1;
}

ur = lsredis_getd( mp->update_resolution );
pos = lspmac_getPosition( mp );

for( i=0; i<plength; i++ ) {
    p = lsredis_get_obj( "%s.presets.%d.position", mp->name,
    i );
    if( fabs( pos - lsredis_getd( p ) ) <= ur ) {
        return i;
    }
}
return -1;
}

```

#### 7.6.3.16 lsredis\_obj\_t\* lsredis\_get\_obj( char \* *fmt*, ... )

Definition at line 596 of file lsredis.c.

```
{
lsredis_obj_t *rtn;
va_list arg_ptr;
char k[512];
char *kp;
int nkp;

va_start( arg_ptr, fmt);
vsnprintf( k, sizeof(k)-1, fmt, arg_ptr);
k[sizeof(k)-1] = 0;
va_end( arg_ptr);

nkp = strlen(k) + strlen( lsredis_head) + 16; // 16
    is overkill. I know. Get over it.
kp = calloc( nkp, sizeof( char));
if( kp == NULL) {
    lslogging_log_message( "lsredis_get_obj: Out of memory
    ");
    exit( -1);
}

snprintf( kp, nkp-1, "%s.%s", lsredis_head, k);
kp[nkp-1] = 0;

pthread_mutex_lock( &lsredis_mutex);
while( lsredis_running == 0)
    pthread_cond_wait( &lsredis_cond, &lsredis_mutex);

rtn = _lsredis_get_obj( kp);
pthread_mutex_unlock( &lsredis_mutex);

free( kp);
return rtn;
}
```

#### 7.6.3.17 double lsredis\_get\_or\_set\_d ( lsredis\_obj\_t \* p, double val, int prec )

Definition at line 360 of file lsredis.c.

```
{
long int rtn;
int err;
struct timespec timeout;

clock_gettime( CLOCK_REALTIME, &timeout);
timeout.tv_sec += 2;

pthread_mutex_lock( &p->mutex);
err = 0;
while( err == 0 && p->valid == 0)
    err = pthread_cond_timedwait( &p->cond, &p->mutex, &timeout);

if( err == ETIMEDOUT) {
    rtn = val;
    pthread_mutex_unlock( &p->mutex);
    lsredis_setstr( p, "%.*f", prec, val);
} else {
    rtn = p->lvalue;
    pthread_mutex_unlock( &p->mutex);
}

return rtn;
}
```

#### 7.6.3.18 long int lsredis\_get\_or\_set\_l ( lsredis\_obj\_t \* p, long int val )

Definition at line 411 of file lsredis.c.

```
{
long int rtn;
int err;
struct timespec timeout;

clock_gettime( CLOCK_REALTIME, &timeout);
timeout.tv_sec += 2;

pthread_mutex_lock( &p->mutex);
```

```

err = 0;
while( err == 0 && p->valid == 0)
    err = pthread_cond_timedwait( &p->cond, &p->mutex, &timeout);

if( err == ETIMEDOUT) {
    lslogging_log_message( "lsredis_get_or_set_l: using
        default value of %ld for redis variable %s", val, p->key);
    rtn = val;
    pthread_mutex_unlock( &p->mutex);
    lsredis_setstr( p, "%ld", val);
} else {
    rtn = p->lvalue;
    pthread_mutex_unlock( &p->mutex);
}

return rtn;
}

```

#### 7.6.3.19 char\*\* lsredis\_get\_string\_array ( lsredis\_obj\_t \* p )

Definition at line 437 of file lsredis.c.

```

{
char **rtn;

pthread_mutex_lock( &p->mutex);
while( p->valid == 0)
    pthread_cond_wait( &p->cond, &p->mutex);

rtn = p->avalue;
pthread_mutex_unlock( &p->mutex);

return rtn;
}

```

#### 7.6.3.20 int lsredis\_getb ( lsredis\_obj\_t \* p )

Definition at line 450 of file lsredis.c.

```

{
int rtn;

pthread_mutex_lock( &p->mutex);
while( p->valid == 0)
    pthread_cond_wait( &p->cond, &p->mutex);

rtn = p->bvalue;
pthread_mutex_unlock( &p->mutex);

return rtn;
}

```

#### 7.6.3.21 char lsredis\_getc ( lsredis\_obj\_t \* p )

Definition at line 463 of file lsredis.c.

```

{
int rtn;

pthread_mutex_lock( &p->mutex);
while( p->valid == 0)
    pthread_cond_wait( &p->cond, &p->mutex);

rtn = p->cvalue;
pthread_mutex_unlock( &p->mutex);

return rtn;
}

```

## 7.6.3.22 double lsredis\_getd ( lsredis\_obj\_t \* p )

Definition at line 385 of file lsredis.c.

```

    {
double rtn;

pthread_mutex_lock( &p->mutex );
while( p->valid == 0 )
    pthread_cond_wait( &p->cond, &p->mutex );

rtn = p->dvalue;
pthread_mutex_unlock( &p->mutex );

return rtn;
}

```

## 7.6.3.23 long int lsredis\_getl ( lsredis\_obj\_t \* p )

Definition at line 398 of file lsredis.c.

```

    {
long int rtn;

pthread_mutex_lock( &p->mutex );
while( p->valid == 0 )
    pthread_cond_wait( &p->cond, &p->mutex );

rtn = p->lvalue;
pthread_mutex_unlock( &p->mutex );

return rtn;
}

```

## 7.6.3.24 char\* lsredis\_getstr ( lsredis\_obj\_t \* p )

return a copy of the key's string value be sure to free the result

Definition at line 284 of file lsredis.c.

```

    {
char *rtn;

// 
// Have to use strdup since we cannot guarantee that p->value won't be freed
// while the caller is still using it
// 
pthread_mutex_lock( &p->mutex );
while( p->valid == 0 )
    pthread_cond_wait( &p->cond, &p->mutex );

rtn = strdup(p->value);
pthread_mutex_unlock( &p->mutex );
return rtn;
}

```

## 7.6.3.25 void lsredis\_hgetCB ( redisAsyncContext \* ac, void \* reply, void \* privdata )

Definition at line 476 of file lsredis.c.

```

    {
redisReply *r;
lsredis_obj_t *p;

r = reply;
p = privdata;

// lslogging_log_message( "hgetCB: %s %s", p == NULL ? "<NULL>" : p->key,
// r->type == REDIS_REPLY_STRING ? r->str : "Non-string value. Why?" );

```

```

// Apparently this item does not exist
// Just set it to an empty string so at least other apps will have the same
// behaviour as us
// TODO: figure out a better way to deal with missing key/values
//
if( p != NULL && r->type == REDIS_REPLY_NIL) {
    lsredis_setstr( p, "");
    return;
}

if( p != NULL && r->type == REDIS_REPLY_STRING && r->str != NULL) {
    pthread_mutex_lock( &p->mutex);

    _lsredis_set_value( p, r->str);

    pthread_cond_signal( &p->cond);
    pthread_mutex_unlock( &p->mutex);
}
}

```

### 7.6.3.26 void lsredis\_init( )

Initialize this module, that is, set up the connections.

#### Parameters

<i>pub</i>	Publish under this (unique) name
<i>re</i>	Regular expression to select keys we want to mirror
<i>head</i>	Prepend this (+ a dot) to the beginning of requested objects

Definition at line 1129 of file lsredis.c.

```

{
int err;

//
// set up hash map to store redis objects
//
err = hcreate_r( 8192, &lsredis_htab);
if( err == 0) {
    lslogging_log_message( "lsredis_init: Cannot create
        hash table. Really bad things are going to happen. hcreate_r returned %d", err);
}

pthread_cond_init( &lsredis_cond, NULL);

subac = redisAsyncConnect("127.0.0.1", 6379);
if( subac->err) {
    lslogging_log_message( "Error: %s", subac->errstr
        );
}

subfd.fd          = subac->c.fd;
subfd.events      = 0;
subac->ev.data   = &subfd;
subac->ev.addRead = lsredis_addRead;
subac->ev.delRead = lsredis_delRead;
subac->ev.addWrite = lsredis_addWrite;
subac->ev.delWrite = lsredis_delWrite;
subac->ev.cleanup = lsredis_cleanup;

roac = redisAsyncConnect("127.0.0.1", 6379);
if( roac->err) {
    lslogging_log_message( "Error: %s", roac->errstr);
}

rofd.fd          = roac->c.fd;
rofd.events      = 0;
roac->ev.data   = &rofd;
roac->ev.addRead = lsredis_addRead;
roac->ev.delRead = lsredis_delRead;
roac->ev.addWrite = lsredis_addWrite;
roac->ev.delWrite = lsredis_delWrite;
roac->ev.cleanup = lsredis_cleanup;

wrac = redisAsyncConnect("127.0.0.1", 6379);

```

```

if( wrac->err) {
    lslogging_log_message( "Error: %s", wrac->errstr);
}

wrfd.fd          = wrac->c.fd;
wrfd.events      = 0;
wrac->ev.data   = &wrfd;
wrac->ev.addRead = lsredis_addRead;
wrac->ev.delRead = lsredis_delRead;
wrac->ev.addWrite = lsredis_addWrite;
wrac->ev.delWrite = lsredis_delWrite;
wrac->ev.cleanup = lsredis_cleanup;

// separate hash table for the presets
//
hcreate_r( lsredis_preset_max_n * 2, &lsredis_preset_ht
    );

pthread_mutex_init( &lsredis_preset_list_mutex, &
    mutex_initializer);
pthread_mutex_init( &lsredis_config_mutex, &
    mutex_initializer);
pthread_cond_init( &lsredis_config_cond, NULL);
}

```

### 7.6.3.27 void lsredis\_keysCB ( redisAsyncContext \* ac, void \* reply, void \* privdata )

Sift through the keys to find ones we like.

Add them to our list of followed objects

Definition at line 807 of file lsredis.c.

```

{
redisReply *r;
int i;

r = reply;
if( r->type != REDIS_REPLY_ARRAY) {
    lslogging_log_message( "lsredis_keysCB: expected
        array...");
    lsredis_debugCB( ac, reply, privdata);
    return;
}

for( i=0; i< (int)r->elements; i++) {
    if( r->element[i]->type != REDIS_REPLY_STRING) {
        lslogging_log_message( "lsredis_keysCB: expected
            string...");
        lsredis_debugCB( ac, r->element[i], privdata);
    } else {
        lsredis_maybe_add_key( r->element[i]->str);
    }
}
}
```

### 7.6.3.28 void lsredis\_load\_presets ( char \* motor\_name )

update the presets hash table for the named motor

Definition at line 830 of file lsredis.c.

```

{
lsredis_obj_t *p;
lsredis_preset_list_t *pl;
int plength;
char *preset_name;
int i;
int key_length;
ENTRY entry_in, *entry_outp;

p = lsredis_get_obj( "%s.presets.length", motor_name);
plength = lsredis_get_or_set_l( p, 0);
if( plength <= 0)
    return;
```

```

pthread_mutex_lock( &lsredis_preset_list_mutex);

for( i=0; i<plength; i++ ) {
    pl = calloc( 1, sizeof( lsredis_preset_list_t));
    pl->name      = lsredis_get_obj( "%s.presets.%d.name",
                                      motor_name, i);
    pl->position   = lsredis_get_obj( "%s.presets.%d.position",
                                      motor_name, i);
    pl->index      = i;

    preset_name    = lsredis_getstr( pl->name);
    key_length     = strlen( motor_name) + strlen( preset_name) + 1;
    pl->key        = calloc( key_length, 1);

    pl->next       = lsredis_preset_list;
    lsredis_preset_list = pl;

    snprintf( pl->key, key_length, "%s%s", motor_name, preset_name);

    entry_in.key   = pl->key;
    entry_in.data  = pl;
    hsearch_r( entry_in, ENTER, &entry_outp, &lsredis_preset_ht
              );
    if( entry_outp->data != pl) {
        //
        // The key was already there or we couldn't add it
        //
        if( entry_outp->data == NULL)
            lslogging_log_message( "lsredis_load_presets:
                                     could not add preset '%s' for motor '%s'", preset_name, motor_name);

        free( pl->key);
        free( pl);
    } else {
        //
        // We've successfully added the new key
        //
        lsredis_preset_n++;
        //
        // Resize the hash table if we are starting to fill it up
        // Generally we prefer a sparse table
        //
        if( lsredis_preset_n >= lsredis_preset_max_n
        ) {
            lslogging_log_message( "lsredis_load_presets:
                                     increasing preset hash table size. max now %d", lsredis_preset_max_n
            );
            hdestroy_r( &lsredis_preset_ht);
            lsredis_preset_max_n *= 2;
            hcreate_r( 2 * lsredis_preset_max_n, &
            lsredis_preset_ht);
            for( pl = lsredis_preset_list; pl != NULL; pl = pl->
next) {
                entry_in.key   = pl->key;
                entry_in.data  = pl;
                hsearch_r( entry_in, ENTER, &entry_outp, &lsredis_preset_ht
                );
            }
            lslogging_log_message( "lsredis_load_presets: done
                                     increasing preset hash table size.", lsredis_preset_max_n
            );
        }
        free( preset_name);
    }
    pthread_mutex_unlock( &lsredis_preset_list_mutex);
}

```

### 7.6.3.29 void lsredis\_maybe\_add\_key ( char \* k )

Definition at line 799 of file lsredis.c.

```

{
if( regexec( &lsredis_key_select_regex, k, 0, NULL, 0
            ) == 0) {
    _lsredis_get_obj( k);
}
}
```

## 7.6.3.30 int lsredis\_reexec ( const regex\_t \* preg, lsredis\_obj\_t \* p, size\_t nmatch, regmatch\_t \* pmatch, int eflags )

Definition at line 267 of file lsredis.c.

```

{
int rtn;

pthread_mutex_lock( &p->mutex );
while( p->valid == 0 )
    pthread_cond_wait( &p->cond, &p->mutex );

rtn = regexec( preg, p->value, nmatch, pmatch, eflags );

pthread_mutex_unlock( &p->mutex );
return rtn;
}

```

## 7.6.3.31 void lsredis\_run ( )

Definition at line 1319 of file lsredis.c.

```

{
pthread_create( &lsredis_thread, NULL, lsredis_worker
    , NULL);
}

```

## 7.6.3.32 void lsredis\_set\_preset ( char \* motor\_name, char \* preset\_name, double dval )

set the given preset to the given value create a new preset if we can't find it

Definition at line 940 of file lsredis.c.

```

{
char s[512];
int plength;
int err;
ENTRY entry_in, *entry_outp;
lsredis_obj_t *p, *presets_length_p;
lsredis_preset_list_t *pl;

snprintf( s, sizeof( s)-1, "%s%s", motor_name, preset_name );
s[sizeof(s)-1] = 0;

entry_in.key = s;
entry_in.data = NULL;
err = hsearch_r( entry_in, FIND, &entry_outp, &lsredis_preset_ht
    );
if( err != 0 ) {
    //
    // Found it. Things are simple.
    //
    pl = entry_outp->data;
    lsredis_setstr( pl->position, "%.3f", dval );
    return;
}
//
// OK, our preset was not found, add it
//
presets_length_p = lsredis_get_obj( "%s.presets.length",
    motor_name );
plength = lsredis_get_or_set_l( presets_length_p, 0 );
plength += 1;

snprintf( s, sizeof( s)-1, "%s.%s.presets.%d.name", lsredis_head,
    motor_name, plength-1 );
s[sizeof(s)-1] = 0;

p = lsredis_get_obj( "%s.presets.%d.name", motor_name, plength
    -1 );
lsredis_setstr( p, "%s", preset_name );

p = lsredis_get_obj( "%s.presets.%d.position", motor_name,
    plength-1 );

```

```

lsredis_setstr( p, "%3f", dval);
lsredis_setstr( presets_length_p, "%ld", plength);
lsredis_load_presets( motor_name);
}

```

### 7.6.3.33 void lsredis\_set\_value ( lsredis\_obj\_t \* p, char \* fmt, ... )

Set the value of a redis object and make it valid.

Called by mgetCB to set the value as it is in redis Maybe TODO: we've arbitrarily set the maximum size of a value here. Although I cannot imagine needed bigger values it would not be a big deal to enable it.

Definition at line 227 of file lsredis.c.

```

{
va_list arg_ptr;
char v[512];

va_start( arg_ptr, fmt);
vsnprintf( v, sizeof(v)-1, fmt, arg_ptr);
va_end( arg_ptr);

v[sizeof(v)-1] = 0;

pthread_mutex_lock( &p->mutex);

_lsredis_set_value( p, v);

pthread_cond_signal( &p->cond);
pthread_mutex_unlock( &p->mutex);
}

```

### 7.6.3.34 void lsredis\_setstr ( lsredis\_obj\_t \* p, char \* fmt, ... )

Set the value and update redis.

Note that lsredis\_set\_value sets the value based on redis while here we set redis based on the value Arbitray maximum string length set here. TODO: Probably this limit should be removed at some point.

redisAsyncCommandArgv used instead of redisAsyncCommand 'cause it's easier (and possible) to deal with strings that would otherwise cause hiredis to emit a bad command, like those containing spaces. < up the count of times we need to see ourselves published before we start listening to others again

< Unlock to prevent deadlock in case the service routine needs to set our value

< redisAsyncCommandArgv shouldn't need to access this after it's made up it's packet (before it returns) so we should be OK with this location disappearing soon.

Definition at line 309 of file lsredis.c.

```

{
va_list arg_ptr;
char v[512];
char *argv[4];

va_start( arg_ptr, fmt);
vsnprintf( v, sizeof(v)-1, fmt, arg_ptr);
v[sizeof(v)-1] = 0;
va_end( arg_ptr);

pthread_mutex_lock( &p->mutex);

// // Don't send an update if a good value has not changed
// if( p->valid && strcmp( v, p->value) == 0) {
//     // nothing to do
//     pthread_mutex_unlock( &p->mutex);
//     return;
// }

```

```

p->wait_for_me++;
pthread_mutex_unlock( &p->mutex);

argv[0] = "HSET";
argv[1] = p->key;
argv[2] = "VALUE";
argv[3] = v;

pthread_mutex_lock( &lsredis_mutex);
while( lsredis_running == 0)
    pthread_cond_wait( &lsredis_cond, &lsredis_mutex);

redisAsyncCommand( wrac, NULL, NULL, "MULTI");
redisAsyncCommandArgv( wrac, NULL, NULL, 4, (const char **)argv, NULL);

redisAsyncCommand( wrac, NULL, NULL, "PUBLISH %s %s", lsredis_publisher
    , p->key);
redisAsyncCommand( wrac, NULL, NULL, "EXEC");
pthread_mutex_unlock( &lsredis_mutex);

// Assume redis will take exactly the value we sent it
//
pthread_mutex_lock( &p->mutex);
_lsredis_set_value( p, v);
pthread_cond_signal( &p->cond);
pthread_mutex_unlock( &p->mutex);
}

```

### 7.6.3.35 void lsredis\_sig\_service ( struct pollfd \* evt )

#### Parameters

in	evt	The pollfd object that triggered this call
----	-----	--

Definition at line 1223 of file lsredis.c.

```

{
struct signalfd_siginfo fdsi;

// Really, we don't care about the signal,
// it's just used to drop out of the poll
// function when there is something for us
// to do.

read( evt->fd, &fdsi, sizeof( struct signalfd_siginfo));
}

```

### 7.6.3.36 void lsredis\_subCB ( redisAsyncContext \* ac, void \* reply, void \* privdata )

Use the publication to request the new value.

Definition at line 707 of file lsredis.c.

```

{
redisReply *r;
lsredis_obj_t *p;
char *k;
char *publisher;
ENTRY htab_input, *htab_output;
int err;

r = (redisReply *)reply;

// Ignore our psubscribe reply
//
if( r->type == REDIS_REPLY_ARRAY && r->elements == 3 && r->element[0]->type
    == REDIS_REPLY_STRING && strcmp( r->element[0]->str, "psubscribe") == 0)
    return;

```

```

// But log other stuff we don't understand
//
if( r->type != REDIS_REPLY_ARRAY ||
    r->elements != 4 ||
    r->element[3]->type != REDIS_REPLY_STRING ||
    r->element[2]->type != REDIS_REPLY_STRING) {

    lslogging_log_message( "lsredis_subCB: unexpected
                           reply");
    lsredis_debugCB( ac, reply, privdata);
    return;
}

//
// Ignore obvious junk
//
k = r->element[3]->str;

if( k == NULL || *k == 0)
    return;

//
// see if we care
//
if( regexec( &lsredis_key_select_regex, k, 0, NULL, 0
            ) == 0) {
    //
    // We should know about this one
    //

    htab_input.key = k;
    htab_input.data = NULL;

    errno = 0;
    err = hsearch_r( htab_input, FIND, &htab_output, &lsredis_htab)
    ;
    if( err == 0 && errno == ESRCH)
        p = NULL;
    else
        p = htab_output->data;

    if( p == NULL) {
        _lsredis_get_obj( k);
    } else {
        // Look who's talk'n
        publisher = r->element[2]->str;

        pthread_mutex_lock( &p->mutex);
        if( p->wait_for_me) {
            //
            // see if we are done waiting
            //
            if( strcmp( publisher, lsredis_publisher)==0)
                p->wait_for_me--;

            pthread_mutex_unlock( &p->mutex);
            //
            // Don't get a new value, either we set it last or we are still waiting
            for redis to report
            // our publication
            //
            return;
        }

        // Here we know our value is out of date
        //
        p->valid = 0;
        lsevents_send_event( "%s Invalid", p->events_name
        );
        pthread_mutex_unlock( &p->mutex);

        //
        // We shouldn't get here if wait_for_me is zero and we are the publisher.
        // If somehow we did (ie we did an hset with out incrementing wait_for_me
        // or if we published too many times), it shouldn't hurt to get the value again.
        //
        redisAsyncCommand( roac, lsredis_hgetCB, p, "HGET %s
                                         VALUE", k);
    }
}
}

```

## 7.6.3.37 void\* lsredis\_worker ( void \* dummy )

subscribe to changes and service sockets

< poll timeout, in millisecs (of course)

< array of pollfd's for the poll function, one entry per connection

< number of active elements in fda

Definition at line 1242 of file lsredis.c.

```
{
static int poll_timeout_ms = -1;
static struct pollfd fda[4];
static int nfda = 0;
static sigset_t our_sigset;
int pollrtn;
int i;

pthread_mutex_lock( &lsredis_mutex );
//  

// block ordinary signal mechanism  

//  

sigemptyset( &our_sigset );
sigaddset( &our_sigset, SIGUSR1 );
pthread_sigmask( SIG_BLOCK, &our_sigset, NULL );

// Set up fd mechanism
//  

fda[0].fd = signalfd( -1, &our_sigset, SFD_NONBLOCK );
if( fda[0].fd == -1 ) {
    char *es;

    es = strerror( errno );
    lslogging_log_message( "lsredis_worker: Signalfd trouble '%s'", es );
}
fda[0].events = POLLIN;
nfda = 1;

lsredis_running = 1;

pthread_cond_signal( &lsredis_cond );
pthread_mutex_unlock( &lsredis_mutex );

while(1) {
    nfda = 1;

    pthread_mutex_lock( &lsredis_mutex );
    if( subfd.fd != -1 ) {
        fda[nfda].fd      = subfd.fd;
        fda[nfda].events  = subfd.events;
        fda[nfda].revents = 0;
        nfda++;
    }

    if( rofd.fd != -1 ) {
        fda[nfda].fd      = rofd.fd;
        fda[nfda].events  = rofd.events;
        fda[nfda].revents = 0;
        nfda++;
    }

    if( wrfd.fd != -1 ) {
        fda[nfda].fd      = wrfd.fd;
        fda[nfda].events  = wrfd.events;
        fda[nfda].revents = 0;
        nfda++;
    }
    pthread_mutex_unlock( &lsredis_mutex );

    pollrtn = poll( fda, nfda, poll_timeout_ms );

    if( pollrtn && fda[0].revents ) {
        lsredis_sig_service( &(fda[0]) );
        pollrtn--;
    }

    for( i=1; i<nfda; i++ ) {
        if( fda[i].revents ) {
            lsredis_fd_service( &(fda[i]) );
        }
    }
}
```

```

    }
}
}
```

### 7.6.3.38 void redisDisconnectCB ( const redisAsyncContext \* ac, int status )

call back in case a redis server becomes disconnected TODO: reconnect

Definition at line 632 of file lsredis.c.

```

if( status != REDIS_OK) {
    lslogging_log_message( "lsredis: Disconnected with
                           status %d", status);
}
}
```

## 7.6.4 Variable Documentation

### 7.6.4.1 pthread\_cond\_t lsredis\_cond

Definition at line 75 of file lsredis.c.

### 7.6.4.2 pthread\_cond\_t lsredis\_config\_cond [static]

Definition at line 91 of file lsredis.c.

### 7.6.4.3 pthread\_mutex\_t lsredis\_config\_mutex [static]

Definition at line 90 of file lsredis.c.

### 7.6.4.4 char\* lsredis\_head = NULL [static]

Definition at line 89 of file lsredis.c.

### 7.6.4.5 struct hsearch\_data lsredis\_htab [static]

Definition at line 81 of file lsredis.c.

### 7.6.4.6 regex\_t lsredis\_key\_select\_regex [static]

Definition at line 88 of file lsredis.c.

### 7.6.4.7 pthread\_mutex\_t lsredis\_mutex = PTHREAD\_RECURSIVE\_MUTEX\_INITIALIZER\_NP

Definition at line 74 of file lsredis.c.

### 7.6.4.8 lsredis\_obj\_t\* lsredis\_objs = NULL [static]

Definition at line 80 of file lsredis.c.

7.6.4.9 **struct hsearch\_data lsredis\_preset\_ht** [static]

Definition at line 106 of file lsredis.c.

7.6.4.10 **lsredis\_preset\_list\_t\* lsredis\_preset\_list = NULL** [static]

Definition at line 105 of file lsredis.c.

7.6.4.11 **pthread\_mutex\_t lsredis\_preset\_list\_mutex** [static]

Definition at line 109 of file lsredis.c.

7.6.4.12 **int lsredis\_preset\_max\_n = 1024** [static]

Definition at line 108 of file lsredis.c.

7.6.4.13 **int lsredis\_preset\_n = 0** [static]

Definition at line 107 of file lsredis.c.

7.6.4.14 **char\* lsredis\_publisher = NULL** [static]

Definition at line 87 of file lsredis.c.

7.6.4.15 **int lsredis\_running = 0**

Definition at line 76 of file lsredis.c.

7.6.4.16 **pthread\_t lsredis\_thread** [static]

Definition at line 72 of file lsredis.c.

7.6.4.17 **pthread\_mutexattr\_t mutex\_initializer** [static]

Definition at line 77 of file lsredis.c.

7.6.4.18 **redisAsyncContext\* roac** [static]

Definition at line 84 of file lsredis.c.

7.6.4.19 **struct pollfd rofd** [static]

Definition at line 94 of file lsredis.c.

7.6.4.20 **redisAsyncContext\* subac** [static]

Definition at line 83 of file lsredis.c.

#### 7.6.4.21 struct pollfd subfd [static]

Definition at line 93 of file lsredis.c.

#### 7.6.4.22 redisAsyncContext\* wrac [static]

Definition at line 85 of file lsredis.c.

#### 7.6.4.23 struct pollfd wrfd [static]

Definition at line 95 of file lsredis.c.

## 7.7 ltest.c File Reference

```
#include "pgpmac.h"
```

### Functions

- void [ltest\\_lspmac\\_est\\_move\\_time\(\)](#)
- void [ltest\\_main\(\)](#)

#### 7.7.1 Function Documentation

##### 7.7.1.1 void ltest\_lspmac\_est\_move\_time( )

Definition at line 14 of file ltest.c.

```
{
int err;
double move_time;
double fudge;
int mmask;
fudge = 2.0;

mmask = 0;
err = lspmac\_est\_move\_time\( &move\_time, &mmask, omega
    , 0, NULL, 360., NULL\);
lslogging\_log\_message\( "ltest\_lspmac\_est\_move\_time:
    omega 360 move\_time=%f err=%d", move\_time, err\);

if( lspmac\_est\_move\_time\_wait\( move\_time + fudge,
    mmask, NULL\)\) {
lslogging\_log\_message\( "ltest\_lspmac\_est\_move\_time:
    timed out"\);
return;
}

err = lspmac\_est\_move\_time\( &move\_time, &mmask, aperz
    , 0, "Cover", 0., NULL\);
lslogging\_log\_message\( "ltest\_lspmac\_est\_move\_time:
    aperz Cover move\_time=%f err=%d", move\_time, err\);

if( lspmac\_est\_move\_time\_wait\( move\_time + fudge,
    mmask, NULL\)\) {
lslogging\_log\_message\( "ltest\_lspmac\_est\_move\_time:
    timed out"\);
return;
}

err = lspmac\_est\_move\_time\( &move\_time, &mmask, aperz
    , 0, "In", 0., NULL\);
lslogging\_log\_message\( "ltest\_lspmac\_est\_move\_time:
    aperz In move\_time=%f err=%d", move\_time, err\);
```

```

if( lspmac_est_move_time_wait( move_time + fudge,
    mmask, NULL) ) {
    lslogging_log_message( "ltest_lspmac_est_move_time:
        timed out");
    return;
}

err = lspmac_est_move_time( &move_time, &mmask, capz,
    0, "Cover", 0., NULL);
lslogging_log_message( "ltest_lspmac_est_move_time:
    capz Cover move_time=%f err=%d", move_time, err);

if( lspmac_est_move_time_wait( move_time + fudge,
    mmask, NULL) ) {
    lslogging_log_message( "ltest_lspmac_est_move_time:
        timed out");
    return;
}

err = lspmac_est_move_time( &move_time, &mmask, capz,
    0, "In", 0., NULL);
lslogging_log_message( "ltest_lspmac_est_move_time:
    capz In move_time=%f err=%d", move_time, err);

if( lspmac_est_move_time_wait( move_time + fudge,
    mmask, NULL) ) {
    lslogging_log_message( "ltest_lspmac_est_move_time:
        timed out");
    return;
}

err = lspmac_est_move_time( &move_time, &mmask, apery
    , 0, "In", 0.0, aperz, 0, "In", 0.0, capy, 0, "In",
    0.0, scint, 0, "Scintillator", 0.0, NULL);
lslogging_log_message( "ltest_lspmac_est_move_time:
    apery In aperz In capy In capz In scint Scintillator move_time=%f err=%d",
    move_time, err);

if( lspmac_est_move_time_wait( move_time + fudge,
    mmask, NULL) ) {
    lslogging_log_message( "ltest_lspmac_est_move_time:
        timed out");
    return;
}

err = lspmac_est_move_time( &move_time, &mmask, apery
    , 0, "In", 0.0, aperz, 0, "Cover", 0.0, capy, 0, "In",
    0.0, capz, 0, "Cover", 0.0, scint, 0, "Cover", 0.0, NULL);
lslogging_log_message( "ltest_lspmac_est_move_time:
    apery Cover aperz Cover capy Cover capz Cover scint Cover move_time=%f err=%d",
    move_time, err);

if( lspmac_est_move_time_wait( move_time + fudge,
    mmask, NULL) ) {
    lslogging_log_message( "ltest_lspmac_est_move_time:
        timed out");
    return;
}

err = lspmac_est_move_time( &move_time, &mmask, apery
    , 1, "In", 0.0, aperz, 1, "In", 0.0, capy, 1, "In",
    0.0, capz, 1, "In", 0.0, scint, 1, "Scintillator", 0.0,
    omega, 0, "manualMount", 0.0, kappa, 0,
    "manualMount", 0.0, NULL);
lslogging_log_message( "ltest_lspmac_est_move_time:
    apery In aperz In capy In capz In scint Scintillator omega manualMount kappa
    Manualmount move_time=%f err=%d", move_time, err);

if( lspmac_est_move_time_wait( move_time + fudge,
    mmask, NULL) ) {
    lslogging_log_message( "ltest_lspmac_est_move_time:
        timed out");
    return;
}

```

### 7.7.1.2 void ltest\_main ( )

Definition at line 92 of file ltest.c.

```
{
```

```
lptest_lspmac_est_move_time();
}
```

## 7.8 Istimer.c File Reference

Support for delayed and periodic events.

```
#include "pgpmac.h"
```

### Data Structures

- struct [Istimer\\_list\\_struct](#)

*Everything we need to know about a timer.*

### Macros

- #define [LSTIMER\\_LIST\\_LENGTH](#) 1024

*We'll allow this many timers. This should be way more than enough.*

- #define [LSTIMER\\_RESOLUTION\\_NSECS](#) 100000

*times within this amount in the future are considered "now" and the events should be called*

### Typedefs

- typedef struct [Istimer\\_list\\_struct](#) [Istimer\\_list\\_t](#)

*Everything we need to know about a timer.*

### Functions

- void [Istimer\\_unset\\_timer](#) (char \*event)

*Unsets all timers for the given event.*

- void [Istimer\\_set\\_timer](#) (char \*event, int shots, unsigned long int secs, unsigned long int nsecs)

*Create a timer.*

- static void [service\\_timers](#) ()

*Send events that are past due, due, or just about to be due.*

- static void [handler](#) (int sig, siginfo\_t \*si, void \*dummy)

*Service the signal.*

- static void \* [Istimer\\_worker](#) (void \*dummy)

*Our worker.*

- void [Istimer\\_init](#) ()

*Initialize the timer list and pthread stuff.*

- void [Istimer\\_run](#) ()

*Start up our thread.*

## Variables

- static int [Istimer\\_active\\_timers](#) = 0
  - count of the number timers we are tracking*
- static [Istimer\\_list\\_t](#) [Istimer\\_list](#) [[LSTIMER\\_LIST\\_LENGTH](#)]
  - Our timer list.*
- static [pthread\\_t](#) [Istimer\\_thread](#)
  - the timer thread*
- static [pthread\\_mutex\\_t](#) [Istimer\\_mutex](#)
  - protect the timer list*
- static [pthread\\_cond\\_t](#) [Istimer\\_cond](#)
  - allows us to be idle when there is nothing to do*
- static [timer\\_t](#) [Istimer\\_timerid](#)
  - our real time timer*
- static int [new\\_timer](#) = 0
  - indicate that a new timer exists and a call to service\_timers is required*

### 7.8.1 Detailed Description

Support for delayed and periodic events.

#### Date

2012

#### Author

Keith Brister

#### Copyright

All Rights Reserved

Definition in file [Istimer.c](#).

### 7.8.2 Macro Definition Documentation

#### 7.8.2.1 #define LSTIMER\_LIST\_LENGTH 1024

We'll allow this many timers. This should be way more than enough.

Definition at line 11 of file [Istimer.c](#).

#### 7.8.2.2 #define LSTIMER\_RESOLUTION\_NSECS 100000

times within this amount in the future are considered "now" and the events should be called

Definition at line 16 of file [Istimer.c](#).

### 7.8.3 Typedef Documentation

#### 7.8.3.1 typedef struct Istimer\_list\_struct Istimer\_list\_t

Everything we need to know about a timer.

## 7.8.4 Function Documentation

### 7.8.4.1 static void handler ( int *sig*, siginfo\_t \* *si*, void \* *dummy* ) [static]

Service the signal.

Definition at line 190 of file lstim.c.

```
{
pthread_mutex_lock( &lstim_mutex );
service_timers();
pthread_mutex_unlock( &lstim_mutex );
}
```

### 7.8.4.2 void lstim\_init ( )

Initialize the timer list and pthread stuff.

Definition at line 270 of file lstim.c.

```
{
int i;
for( i=0; i<LSTIMER_LIST_LENGTH; i++ ) {
    lstim_list[i].shots = 0;
}

pthread_mutex_init( &lstim_mutex, NULL );
pthread_cond_init( &lstim_cond, NULL );
}
```

### 7.8.4.3 void lstim\_run ( )

Start up our thread.

Definition at line 284 of file lstim.c.

```
{
pthread_create( &lstim_thread, NULL, lstim_worker
                , NULL );
}
```

### 7.8.4.4 void lstim\_set\_timer ( char \* *event*, int *shots*, unsigned long int *secs*, unsigned long int *nsecs* )

Create a timer.

#### Parameters

<i>event</i>	Name of the event to send when the timer goes off
<i>shots</i>	Number of times to run. 0 means never, -1 means forever
<i>secs</i>	Number of seconds to wait
<i>nsecs</i>	Number of nano-seconds to run in addition to secs

Definition at line 63 of file lstim.c.

```
{
int i;
struct timespec now;

// Time we were called. Delay is based on call time, not queued time
//
```

```

clock_gettime( CLOCK_REALTIME, &now);

// Make sure our event is registered (saves a tiny bit of time later)
// lsevents_preregister_event( event);

pthread_mutex_lock( &lstimer_mutex);

for( i=0; i<LSTIMER_LIST_LENGTH; i++) {
    if( lstimer_list[i].shots == 0)
        break;
}

if( i == LSTIMER_LIST_LENGTH) {
    pthread_mutex_unlock( &lstimer_mutex);

    lslogging_log_message( "lstimer_set_timer: out of
                           timers for event: %s, shots: %d, secs: %u,
                           nsecs: %u",
                           event, shots, secs, nsecs);
    return;
}

strncpy( lstimer_list[i].event, event, LSEVENTS_EVENT_LENGTH
         - 1);
lstimer_list[i].event[LSEVENTS_EVENT_LENGTH
         - 1] = 0;
lstimer_list[i].shots      = shots;
lstimer_list[i].delay_secs  = secs;
lstimer_list[i].delay_nsecs = nsecs;

lstimer_list[i].next_secs   = secs + now.tv_sec + (
    now.tv_nsec + nsecs) / 1000000000;
lstimer_list[i].next_nsecs = (now.tv_nsec + nsecs
    ) % 1000000000;
lstimer_list[i].last_secs   = 0;
lstimer_list[i].last_nsecs = 0;

lstimer_list[i].ncalls      = 0;
lstimer_list[i].init_secs    = now.tv_sec;
lstimer_list[i].init_nsecs  = now.tv_nsec;

if( shots != 0) {
    lstimer_active_timers++;
    new_timer++;
}

pthread_cond_signal( &lstimer_cond);
pthread_mutex_unlock( &lstimer_mutex);
}

```

#### 7.8.4.5 void Istimer\_unset\_timer( char \* event )

Unsets all timers for the given event.

Definition at line 46 of file Istimer.c.

```

{
int i;

for( i=0; i<LSTIMER_LIST_LENGTH; i++) {
    if( strcmp( event, lstimer_list[i].event) == 0) {
        lstimer_list[i].shots = 0;
    }
}
}
```

#### 7.8.4.6 static void\* Istimer\_worker( void \* dummy ) [static]

Our worker.

The main loop runs when a new timer is added. The service routine deals with maintenance.

##### Parameters

in	<i>dummy</i>	required by protocol
----	--------------	----------------------

Definition at line 200 of file lstimerc.c.

```

{
    struct sigevent sev;
    struct sigaction sa;
    sigset_t mask;

    // See example at
    // http://www.kernel.org/doc/man-pages/online/pages/man2/timer_create.2.html
    //

    // Set up handler
    //
    sa.sa_flags = SA_SIGINFO;
    sa.sa_sigaction = handler;
    sigemptyset(&sa.sa_mask);
    if (sigaction(SIGRTMIN, &sa, NULL) == -1) {
        lslogging_log_message("lstimerc_worker: sigaction failed");
        exit(-1);
    }

    // Create the timer
    //
    sev.sigev_notify = SIGEV_SIGNAL;
    sev.sigev_signo = SIGRTMIN;
    sev.sigev_value.sival_ptr = &lstimerc_timerid;
    timer_create(CLOCK_REALTIME, &sev, &lstimerc_timerid);

    // Block timer signal for now since we really
    // want to be sure we do not own a lock on the timer mutex
    // while servicing the signal
    //
    sigemptyset(&mask);
    sigaddset(&mask, SIGRTMIN);

    while( 1 ) {
        pthread_mutex_lock( &lstimerc_mutex );

        while( new_timer == 0 )
            pthread_cond_wait( &lstimerc_cond, &lstimerc_mutex );
        //

        // ignore signals so we don't service the signal while we are already in
        // the
        // service routine
        //
        sigprocmask( SIG_SETMASK, &mask, NULL );

        //
        // Setting up the timer interval is in the handler
        // so just call it
        //
        service_timers();

        //
        // Reset our flag
        //
        new_timer = 0;

        pthread_mutex_unlock( &lstimerc_mutex );

        // Let the signals rain down
        //
        sigprocmask( SIG_UNBLOCK, &mask, NULL );
    }
}

```

#### 7.8.4.7 static void service\_timers( ) [static]

Send events that are past due, due, or just about to be due.

Definition at line 118 of file lstimerc.c.

```

int
i,
found_active;
{
```

```

lstimerc_list_t *p;
struct timespec now, then, soonest;
struct itimerspec its;

// Did I remind you not to let this thread own the lstimer mutex outside of
// this
// service routine when SIGRTMIN is active?
//

// Call with lstimer_mutex locked

clock_gettime( CLOCK_REALTIME, &now);
//
// Project a tad into the future
then.tv_sec = now.tv_sec + (now.tv_nsec + LSTIMER_RESOLUTION_NSECS
    ) / 1000000000;
then.tv_nsec = (now.tv_nsec + LSTIMER_RESOLUTION_NSECS
    ) % 1000000000;

found_active = 0;
for( i=0; i<lstimerc_active_timers; i++) {
    p = &(lstimerc_list[i]);
    if( p->shots != 0) {
        found_active++;
        if( p->next_secs < then.tv_sec || (p->next_secs ==
            then.tv_sec && p->next_nsecs <= then.tv_nsec)) {
            lsevents_send_event( p->event);
            //
            // After sending the event, compute the next time we need to do this
            //
            p->last_secs = now.tv_sec;
            p->last_nsecs = now.tv_nsec;
            p->ncalls++;
            //
            // Decrement non-infinite loops
            if( p->shots != -1)
                p->shots--;
            if( p->shots == 0) {
                //
                // Take this timer out of the mix
                lstimer_active_timers--;
            } else {
                p->next_secs = p->init_secs + (p->ncalls+1)
                    * p->delay_secs + (p->init_nsecs + (p->ncalls+1)*p->
                    delay_nsecs)/1000000000;
                p->next_nsecs = (p->init_nsecs + (p->ncalls
                    +1)*p->delay_nsecs) % 1000000000;
            }
        }
    }

    if( found_active == 1) {
        soonest.tv_sec = p->next_secs;
        soonest.tv_nsec = p->next_nsecs;
    } else {
        if( soonest.tv_sec > p->next_secs || (soonest.tv_sec == p->
            next_secs && soonest.tv_nsec > p->next_nsecs)) {
            soonest.tv_sec = p->next_secs;
            soonest.tv_nsec = p->next_nsecs;
        }
    }
}

if( soonest.tv_sec != 0) {
    its.it_value.tv_sec = soonest.tv_sec;
    its.it_value.tv_nsec = soonest.tv_nsec;
    its.it_interval.tv_sec = 0;
    its.it_interval.tv_nsec = 0;
    timer_settime( lstimer_timerid, TIMER_ABSTIME, &its, NULL);
}
}

```

## 7.8.5 Variable Documentation

### 7.8.5.1 int lstimer\_active\_timers = 0 [static]

count of the number timers we are tracking

Definition at line 18 of file lstimer.c.

### 7.8.5.2 `pthread_cond_t lstimber_cond [static]`

allows us to be idle when there is nothing to do

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

### 7.8.5.3 `lstimber_list_t lstimber_list[LSTIMER_LIST_LENGTH] [static]`

Our timer list.

Definition at line 36 of file `lstimber.c`.

### 7.8.5.4 `pthread_mutex_t lstimber_mutex [static]`

protect the timer list

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

### 7.8.5.5 `pthread_t lstimber_thread [static]`

the timer thread

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

### 7.8.5.6 `timer_t lstimber_timerid [static]`

our real time timer

Definition at line 41 of file `lstimber.c`.

### 7.8.5.7 `int new_timer = 0 [static]`

indicate that a new timer exists and a call to `service_timers` is required

Definition at line 42 of file `lstimber.c`.

## 7.9 md2cmds.c File Reference

Implements commands to run the md2 diffractometer attached to a PMAC controled by postgresql.

```
#include "pgpmac.h"
```

### Data Structures

- struct `md2cmds_cmd_kv_struct`

### TypeDefs

- typedef struct  
`md2cmds_cmd_kv_struct` `md2cmds_cmd_kv_t`

## Functions

- int `md2cmds_abort` (const char \*dummy)  
*abort the current motion and put the system into a known state /param dummy Unused here*
- int `md2cmds_collect` (const char \*dummy)  
*Collect some data.*
- int `md2cmds_moveAbs` (const char \*ccmd)  
*Move a motor to the position requested Returns non zero on error.*
- int `md2cmds_moveRel` (const char \*ccmd)  
*Move a motor to the position requested Returns non zero on error.*
- int `md2cmds_phase_change` (const char \*ccmd)  
*Move md2 devices to a preconfigured state.*
- int `md2cmds_run_cmd` (const char \*)
- int `md2cmds_rotate` (const char \*dummy)  
*Spin 360 and make a video (recenter first, maybe)*
- int `md2cmds_set` (const char \*)
- int `md2cmds_settransferpoint` (const char \*)
- int `md2cmds_test` (const char \*dummy)  
*Run the test routine(s)*
- int `md2cmds_transfer` (const char \*dummy)  
*Transfer a sample.*
- void `md2cmds_push_queue` (char \*action)
- void `md2cmds_home_prep` ()
- int `md2cmds_home_wait` (double timeout\_secs)
- void `md2cmds_move_prep` ()  
*prepare for new movements*
- int `md2cmds_move_wait` (double timeout\_secs)  
*Wait for all the motions requested to complete.*
- int `md2cmds_is_moving` ()  
*returns non-zero if we think a motor is moving, 0 otherwise*
- double `md2cmds_prep_axis` (lspmac\_motor\_t \*mp, double pos)
- void `md2cmds_organs_move_presets` (char \*pay, char \*paz, char \*pcy, char \*pcz, char \*psz)
- int `md2cmds_phase_manualMount` ()  
*Go to the manual mount phase.*
- int `md2cmds_phase_robotMount` ()  
*Go to robot mount phase.*
- int `md2cmds_phase_center` ()  
*Go to center phase.*
- int `md2cmds_phase_dataCollection` ()  
*Go to data collection phase.*
- int `md2cmds_phase_beamLocation` ()  
*Go to beam location phase.*
- int `md2cmds_phase_safe` ()  
*Go to safe phase.*
- void `md2cmds_mvcenter_move` (double cx, double cy, double ax, double ay, double az)  
*Move the centering and alignment tables.*
- void `md2cmds_maybe_done_moving_cb` (char \*event)  
*Track how many motors are moving.*
- void `md2cmds_maybe_done_homing_cb` (char \*event)  
*Track motors homing.*
- void `md2cmds_kappaphi_move` (double kappa\_deg, double phi\_deg)
- void `md2cmds_rotate_cb` (char \*event)

- void `md2cmds_maybe_rotate_done_cb` (char \*event)
 

*Tell the database about the time we went through omega=zero.*
- void `md2cmds_set_scale_cb` (char \*event)
 

*Now that we are done with the 360 rotation lets rehome right quick.*
- void `md2cmds_time_capz_cb` (char \*event)
 

*Fix up xscale and yscale when zoom changes xscale and yscale have units of microns per pixel.*
- void `md2cmds_action_queue` (double timeout, char \*action)
 

*Time the capillary motion for the transfer routine.*
- void `md2cmds_action_wait` ()
 

*pause until md2cmds\_worker has finished running the command*
- void \* `md2cmds_worker` (void \*dummy)
 

*Our worker thread.*

  - void `md2cmds_coordsys_1_stopped_cb` (char \*event)
  - void `md2cmds_coordsys_2_stopped_cb` (char \*event)
  - void `md2cmds_coordsys_3_stopped_cb` (char \*event)
  - void `md2cmds_coordsys_4_stopped_cb` (char \*event)
  - void `md2cmds_coordsys_5_stopped_cb` (char \*event)
  - void `md2cmds_coordsys_7_stopped_cb` (char \*event)
  - void `md2cmds_init` ()
 

*Initialize the md2cmds module.*
- void `md2cmds_run` ()
 

*Start up the thread.*

## Variables

- pthread\_cond\_t `md2cmds_cond`

*condition to signal when it's time to run an md2 command*
- pthread\_mutex\_t `md2cmds_mutex`

*mutex for the condition*
- int `md2cmds_moving_queue_wait` = 0
- pthread\_cond\_t `md2cmds_moving_cond`

*wait for command to have been dequeued and run*
- pthread\_mutex\_t `md2cmds_moving_mutex`

*message passing between md2cmds and pg*
- int `md2cmds_homing_count` = 0
 

*We've asked a motor to home.*
- pthread\_cond\_t `md2cmds_homing_cond`

*coordinate homing and homed*
- pthread\_mutex\_t `md2cmds_homing_mutex`

*our mutex;*
- int `md2cmds_moving_count` = 0
- char `md2cmds_cmd` [MD2CMDS\_CMD\_LENGTH]
 

*our command;*
- lsredis\_obj\_t \* `md2cmds_md_status_code`
- static pthread\_t `md2cmds_thread`
- static int `rotating` = 0
 

*flag: when omega is in position after a rotate we want to re-home omega*
- static double `md2cmds_capz_moving_time` = NAN
- static struct hsearch\_data `md2cmds_hmap`
- static regex\_t `md2cmds_cmd_regex`
- static `md2cmds_cmd_kv_t` `md2cmds_cmd_kvs` []
 

*static*
- static `md2cmds_cmd_kv_t` `md2cmds_cmd_pg_kvs` []
 

*static*

### 7.9.1 Detailed Description

Implements commands to run the md2 diffractometer attached to a PMAC controled by postgresql.

#### Date

2012

#### Author

Keith Brister

#### Copyright

All Rights Reserved

Definition in file [md2cmds.c](#).

### 7.9.2 Typedef Documentation

#### 7.9.2.1 `typedef struct md2cmds_cmd_kv_struct md2cmds_cmd_kv_t`

### 7.9.3 Function Documentation

#### 7.9.3.1 `int md2cmds_abort( const char * dummy )`

abort the current motion and put the system into a known state /param dummy Unused here

Definition at line 1602 of file md2cmds.c.

```
{
// First priority is to close the shutter
//
if( fshut->moveAbs( fshut, 0))
    lslogging_log_message( "md2cmds_abort: for some reason
        the shutter close requested failed. Proceeding anyway.");

//
// Now stop all the motors
//
lspmac_abort();
if( md2cmds_move_wait( 10.0))
    lslogging_log_message( "md2cmds_abort: Some motors did
        not appear to stop. Proceeding with reset anyway");

//
// Now try to close the shutter (again)
//
if( fshut->moveAbs( fshut, 0))
    lslogging_log_message( "md2cmds_abort: for some reason
        the shutter close requested failed (2). Proceeding anyway.");

//
// Force the motion flags down
//
lspmac_SockSendDPLine( NULL, "m5075=0");

return 0;
}
```

#### 7.9.3.2 `int md2cmds_action_queue( double timeout, char * action )`

Definition at line 1564 of file md2cmds.c.

```

{
int rtn;
struct timespec waitforit;

if( timeout < 0.0) {
    rtn = pthread_mutex_lock( &md2cmds_mutex);
} else {
    clock_gettime( CLOCK_REALTIME, &waitforit);

    waitforit.tv_sec += floor(timeout);

    waitforit.tv_nsec += (timeout - waitforit.tv_sec)*1.e9;
    while( waitforit.tv_nsec >= 1000000000) {
        waitforit.tv_sec++;
        waitforit.tv_nsec -= 1000000000;
    }

    rtn = pthread_mutex_timedlock( &md2cmds_mutex, &waitforit);
}

if( rtn == 0) {
    strncpy( md2cmds_cmd, action, MD2CMDS_CMD_LENGTH
            -1);
    md2cmds_cmd[MD2CMDS_CMD_LENGTH-1] = 0;
    pthread_cond_signal( &md2cmds_cond);
    pthread_mutex_unlock( &md2cmds_mutex);
} else {
    if( rtn == ETIMEDOUT)
        lslogging_log_message( "md2cmds_action_queue: %s not
                               queued, operation timed out", action);
    else
        lslogging_log_message( "md2cmds_action_queue: %s not
                               queued with error code %d", action, rtn);
}
return rtn;
}

```

### 7.9.3.3 void md2cmds\_action\_wait( )

pause until md2cmds\_worker has finished running the command

Definition at line 1632 of file md2cmds.c.

```

{
pthread_mutex_lock( &md2cmds_mutex);
pthread_mutex_unlock( &md2cmds_mutex);
}

```

### 7.9.3.4 int md2cmds\_collect( const char \* dummy )

Collect some data.

#### Parameters

<i>dummy</i>	Unused returns non-zero on error
--------------	----------------------------------

< index of shot to be taken  
< Exposure time (saved to compute shutter timeout)  
< start cnts  
< delta cnts  
< omega velocity cnts/msec  
< acceleration time (msec)  
< exposure time (msec)  
< unit to counts conversion

< nominal zero offset  
< maximum acceleration allowed for omega  
< current kappa position in case we need to move phi only  
< current phi position in case we need to move kappa only  
< setup timeouts for shutter

Definition at line 1042 of file md2cmds.c.

```

long long skey;
double exp_time;
double p170;
double p171;
double p173;
double p175;
double p180;
double u2c;
double neutral_pos;
double max_accel;
double kappa_pos;
double phi_pos;
struct timespec now, timeout;
int err;
double move_time;
int mmask;

lsevents_send_event( "Data Collection Starting");

u2c      = lsredis_getd( omega->u2c);
neutral_pos = lsredis_getd( omega->neutral_pos);
max_accel = lsredis_getd( omega->max_accel);

mmask = 0;
err = lspmac_est_move_time( &move_time, &mmask,
                           apery,      1, "In",     0.0, // Aperture to
                           the In position
                           aperz,      1, "In",     0.0,
                           capy,       1, "In",     0.0, // Capillary /
                           Beamstop to the In position
                           capz,       1, "In",     0.0,
                           scint,      1, "Cover",  0.0, // Hide the
                           scintillator
                           blight_ud, 1, NULL,    0.0, // put
                           the backlight down
                           NULL);

err = lspmac_est_move_time_wait( move_time + 2.0,
                                 mmask, NULL);
if( err) {
  lsevents_send_event( "Data Collection Aborted");
  return 1;
}

// // reset shutter has opened flag
// // lspmac_SockSendDpline( NULL, "P3001=0 P3002=0");

while( 1) {
  lsgp_nextshot_call();
  lsgp_nextshot_wait();

  exp_time = lsgp_nextshot.dsexp;

  if( lsgp_nextshot.no_rows_returned) {
    lsgp_nextshot_done();
    break;
  }

  skey = lsgp_nextshot.skey;
  lsgp_query_push( NULL, "SELECT px.shots_set_state(%lld,
  'Preparing')", skey);

  if( lsgp_nextshot.active) {
    if(
      //
      // Don't move if we are within 0.1 microns of our destination
      //
      (fabs( lsgp_nextshot.cx - cenx->position) >
      0.1) ||
      
```

```

(fabs( lsgpg_nextshot.cy - ceny->position) >
0.1) ||
(fabs( lsgpg_nextshot.ax - alignx->position
) > 0.1) ||
(fabs( lsgpg_nextshot.ay - aligny->position
) > 0.1) ||
(fabs( lsgpg_nextshot.az - alignz->position
) > 0.1)) {

lslogging_log_message( "md2cmds_collect: moving
center to cx=%f, cy=%f, ax=%f, ay=%f",lsgpg_nextshot.cx,
lsgpg_nextshot.cy, lsgpg_nextshot.ax, lsgpg_nextshot
.ay, lsgpg_nextshot.az);

err = lspmac_est_move_time( &move_time, &mmask,
                           cenx, 0, NULL, lsgpg_nextshot
                           .cx,
                           ceny, 0, NULL, lsgpg_nextshot
                           .cy,
                           alignx, 0, NULL, lsgpg_nextshot
                           .ax,
                           aligny, 0, NULL, lsgpg_nextshot
                           .ay,
                           alignz, 0, NULL, lsgpg_nextshot
                           .az,
                           NULL);
if( err) {
    lsevents_send_event( "Data Colection Aborted");
    return 1;
}

err = lspmac_est_move_time_wait( move_time,
mmask, NULL);
if( err) {
    lsevents_send_event( "Data Colection Aborted");
    lsgpg_query_push( NULL, "SELECT
px.unlock_diffractometer()");
    return 1;
}
}

// Maybe move kappa and/or phi
//
if( !lsgpg_nextshot.dsphi_isnull || !lsgpg_nextshot
.ds kappa_isnull) {

kappa_pos = lsgpg_nextshot.ds kappa_isnull ?
lspmac_getPosition( kappa) : lsgpg_nextshot.
ds kappa;
phi_pos = lsgpg_nextshot.dsphi_isnull ?
lspmac_getPosition( phi) : lsgpg_nextshot.
dsphi;

lslogging_log_message( "md2cmds_collect: move
phy/kappa: kappa=%f phi=%f", kappa_pos, phi_pos);

err = lspmac_est_move_time( &move_time,
                           kappa, 0, NULL, kappa_pos,
                           phi, 0, NULL, phi_pos,
                           NULL);
if( err) {
    lsgpg_query_push( NULL, "SELECT px.shots_set_state(%lld,
'Error')", skey);
    lsevents_send_event( "Data Colection Aborted");
    return 1;
}

err = lspmac_est_move_time_wait( move_time + 2,
mmask, NULL);
if( err) {
    lsgpg_query_push( NULL, "SELECT px.shots_set_state(%lld,
'Error')", skey);
    lsevents_send_event( "Data Colection Aborted");
    return 1;
}

// 
// Calculate the parameters we'll need to run the scan
//
p180 = lsgpg_nextshot.ds exp * 1000.0;
p170 = u2c * (lsgpg_nextshot.sstart + neutral_pos);
p171 = u2c * lsgpg_nextshot.dsowidth;
p173 = fabs(p180) < 1.e-4 ? 0.0 : u2c * lsgpg_nextshot.dsowidth

```

```

    / p180;
    p175 = p173/max_accel;

    //
    // free up access to nextshot
    //
    lspg_nextshot_done();

    //
    // prepare the database and detector to expose
    // On exit we own the diffractometer lock and
    // have checked that all is OK with the detector
    //
    lspg_seq_run_prep_all( skey,
                           kappa->position,
                           phi->position,
                           cenx->position,
                           ceny->position,
                           alignx->position,
                           aligny->position,
                           alignz->position
                           );

    //
    // make sure our opened flag is down
    // wait for the p3001=0 command to be noticed
    //
    clock_gettime( CLOCK_REALTIME, &now);
    timeout.tv_sec  = now.tv_sec + 10;
    timeout.tv_nsec = now.tv_nsec;

    err = 0;
    pthread_mutex_lock( &lspmac_shutter_mutex);
    while( err == 0 && lspmac_shutter_has_opened != 0)
        err = pthread_cond_timedwait( &lspmac_shutter_cond, &
                                     lspmac_shutter_mutex, &timeout);
    pthread_mutex_unlock( &lspmac_shutter_mutex);

    if( err == ETIMEDOUT) {
        pthread_mutex_unlock( &lspmac_shutter_mutex);
        lslogging_log_message( "md2cmds_collect: Timed out
                               waiting for shutter to be confirmed closed. Data collection aborted.");
        lspg_query_push( NULL, "SELECT px.shots_set_state(%lld,
'Error')", skey);
        lspg_query_push( NULL, "SELECT px.unlock_diffractometer()"
                     );
        lsevents_send_event( "Data Collection Aborted");
        return 1;
    }

    //
    // Start the exposure
    //
    lspmac_set_motion_flags( &mmask, omega);
    lspmac_SockSendDPLine( "Exposure",
                           "&1 P170=%1f P171=%1f P173=%1f P174=0 P175=%1f
                           P176=0 P177=1 P178=0 P180=%1f M431=1 &1B131R",
                           p170,           p171,           p173,           p175,
                           p180);

    //
    // We could look for the "Exposure command accepted" event at this point.
    //
    //
    // wait for the shutter to open
    //
    clock_gettime( CLOCK_REALTIME, &now);
    timeout.tv_sec  = now.tv_sec + 10;
    timeout.tv_nsec = now.tv_nsec;

    err = 0;
    pthread_mutex_lock( &lspmac_shutter_mutex);
    while( err == 0 && lspmac_shutter_has_opened == 0)
        err = pthread_cond_timedwait( &lspmac_shutter_cond, &
                                     lspmac_shutter_mutex, &timeout);

    if( err == ETIMEDOUT) {
        pthread_mutex_unlock( &lspmac_shutter_mutex);
        lslogging_log_message( "md2cmds_collect: Timed out
                               waiting for shutter to open. Data collection aborted.");
        lspg_query_push( NULL, "SELECT px.unlock_diffractometer()"
                     );
        lspg_query_push( NULL, "SELECT px.shots_set_state(%lld,
'Error')", skey);
        lsevents_send_event( "Data Collection Aborted");
    }

```

```

    return 1;
}

/*
// wait for the shutter to close
//
clock_gettime( CLOCK_REALTIME, &now);
lslogging_log_message( "md2cmds_collect: waiting %f
seconds for the shutter to close", 4 + exp_time);
timeout.tv_sec = now.tv_sec + 4 + ceil(exp_time);           // hopefully 4
seconds is long enough to never catch a legitimate shutter close and short
enough to bail when something is really wrong
timeout.tv_nsec = now.tv_nsec;

err = 0;
while( err == 0 && lspmac_shutter_state == 1)
    err = pthread_cond_timedwait( &lspmac_shutter_cond, &
        lspmac_shutter_mutex, &timeout);
pthread_mutex_unlock( &lspmac_shutter_mutex);

if( err == ETIMEDOUT) {
    pthread_mutex_unlock( &lspmac_shutter_mutex);
    lspg_query_push( NULL, "SELECT px.unlock_diffractometer()");
}
lspg_query_push( NULL, "SELECT px.shots_set_state(%lld,
'Error')", skey);
lslogging_log_message( "md2cmds_collect: Timed out
waiting for shutter to close. Data collection aborted.");
lsevents_send_event( "Data Collection Aborted");
return 1;
}

/*
// Signal the detector to start reading out
//
lspg_query_push( NULL, "SELECT px.unlock_diffractometer()");

//
// Update the shot status
//
lspg_query_push( NULL, "SELECT px.shots_set_state(%lld,
'Writing')", skey);

//
// reset shutter has opened flag
//
lspmac_SockSendDPLine( NULL, "P3001=0");

//
// Wait for omega to stop moving
//
if( md2cmds_move_wait( 10.0) {
    lslogging_log_message( "md2cmds_collect: Giving up
waiting for omega to stop moving. Data collection aborted.");
    lsevents_send_event( "Data Collection Aborted");
    return 1;
}

//
// Move the center/alignment stages to the next position
//
// TODO: position omega for the next shot. During data collection the
// motion program
// makes a good guess but for ortho snaps it is wrong. We should add an
// argument to the motion program
//

if( !lspg_nextshot.active2_isnull &&
    lspg_nextshot.active2) {
    if(
        (fabs( lspg_nextshot.cx2 - cenx->position)
        > 0.1) ||
        (fabs( lspg_nextshot.cy2 - ceny->position)
        > 0.1) ||
        (fabs( lspg_nextshot.ax2 - alignx->position
        ) > 0.1) ||
        (fabs( lspg_nextshot.ay2 - aligny->position
        ) > 0.1) ||
        (fabs( lspg_nextshot.az2 - alignz->position
        ) > 0.1)) {

            md2cmds_move_prep();
            md2cmds_mvcenter_move( lspg_nextshot.

```

```
    cx, lsgp_nextshot.cy, lsgp_nextshot.ax,
    lsgp_nextshot.ay, lsgp_nextshot.az);
}
lsevents_send_event( "Data Collection Done");
return 0;
}
```

#### 7.9.3.5 void md2cmds\_coorsys\_1\_stopped\_cb ( char \* event )

Definition at line 1869 of file md2cmds.c.

```
{
}
```

#### 7.9.3.6 void md2cmds\_coorsys\_2\_stopped\_cb ( char \* event )

Definition at line 1871 of file md2cmds.c.

```
{
}
```

#### 7.9.3.7 void md2cmds\_coorsys\_3\_stopped\_cb ( char \* event )

Definition at line 1873 of file md2cmds.c.

```
{
}
```

#### 7.9.3.8 void md2cmds\_coorsys\_4\_stopped\_cb ( char \* event )

Definition at line 1875 of file md2cmds.c.

```
{
}
```

#### 7.9.3.9 void md2cmds\_coorsys\_5\_stopped\_cb ( char \* event )

Definition at line 1877 of file md2cmds.c.

```
{
}
```

#### 7.9.3.10 void md2cmds\_coorsys\_7\_stopped\_cb ( char \* event )

Definition at line 1879 of file md2cmds.c.

```
{
}
```

### 7.9.3.11 void md2cmds\_home\_prep( )

Definition at line 91 of file md2cmds.c.

```
{
pthread_mutex_lock( &md2cmds_homing_mutex);
md2cmds_homing_count = -1;
pthread_mutex_unlock( &md2cmds_homing_mutex);
}
```

### 7.9.3.12 int md2cmds\_home\_wait( double timeout\_secs )

Definition at line 98 of file md2cmds.c.

```
{
struct timespec timeout, now;
double isecs, fsecs;
int err;

clock_gettime( CLOCK_REALTIME, &now);

fsecs = modf( timeout_secs, &isecs);

timeout.tv_sec = now.tv_sec + (long)floor( isecs);
timeout.tv_nsec = now.tv_nsec + (long)floor( fsecs * 1.0e9);

timeout.tv_sec += timeout.tv_nsec / 1000000000;
timeout.tv_nsec %= 1000000000;

err = 0;
pthread_mutex_lock( &md2cmds_homing_mutex);
while( err == 0 && md2cmds_homing_count == -1)
    err = pthread_cond_timedwait( &md2cmds_homing_cond, &
        md2cmds_homing_mutex, &timeout);

if( err != 0) {
    if( err != ETIMEDOUT) {
        lslogging_log_message( "md2cmds_home_wait:
            unexpected error from timedwait: %d tv_sec %ld tv_nsec %ld", err, timeout.tv_sec,
            timeout.tv_nsec);
    }
    pthread_mutex_unlock( &md2cmds_homing_mutex);
    return 1;
}

err = 0;
while( err == 0 && md2cmds_homing_count > 0)
    err = pthread_cond_timedwait( &md2cmds_homing_cond, &
        md2cmds_homing_mutex, &timeout);
pthread_mutex_unlock( &md2cmds_homing_mutex);

if( err != 0) {
    if( err != ETIMEDOUT)
        lslogging_log_message( "md2cmds_home_wait:
            unexpected error from timedwait: %d", err);

    return 1;
}
return 0;
}
```

### 7.9.3.13 void md2cmds\_init( )

Initialize the md2cmds module.

Definition at line 1885 of file md2cmds.c.

```
{
ENTRY hloader, *h rtnval;
int i, err;
int ncmds;

pthread_mutexattr_t mutex_initializer;

pthread_mutexattr_init( &mutex_initializer);
```

```

pthread_mutexattr_settype( &mutex_initializer, PTHREAD_MUTEX_RECURSIVE);

pthread_mutex_init( &md2cmds_mutex, &mutex_initializer);
pthread_cond_init( &md2cmds_cond, NULL);

pthread_mutex_init( &md2cmds_moving_mutex, &
    mutex_initializer);
pthread_cond_init( &md2cmds_moving_cond, NULL);

pthread_mutex_init( &md2cmds_homing_mutex, &
    mutex_initializer);
pthread_cond_init( &md2cmds_homing_cond, NULL);

err = regcomp( &md2cmds_cmd_regex, " *([^\n]+) (([^\n]+)\\\
    .presets\\..)*([^\n]*)([^\n]*)", REG_EXTENDED);
if( err != 0) {
    int nerrmsg;
    char *errmsg;

    nerrmsg = regerror( err, &md2cmds_cmd_regex, NULL, 0);
    if( nerrmsg > 0) {
        errmsg = calloc( nerrmsg, sizeof( char));
        nerrmsg = regerror( err, &md2cmds_cmd_regex, errmsg,
        nerrmsg);
        lslogging_log_message( "md2cmds_init: %s", errmsg);
        free( errmsg);
    }
}

md2cmds_md_status_code = lsredis_get_obj
    ( "md2_status_code");
lsredis_setstr( md2cmds_md_status_code, "
    7");

ncmds = sizeof(md2cmds_cmd_kvs)/sizeof(md2cmds_cmd_kvs
[0]);
if( pgpmac_use_pg) {
    ncmds += sizeof(md2cmds_cmd_pg_kvs)/sizeof(
        md2cmds_cmd_pg_kvs[0]);
}

hcreate_r( 2 * ncmds, &md2cmds_hmap);

for( i=0; i<sizeof(md2cmds_cmd_kvs)/sizeof(md2cmds_cmd_kvs
[0]); i++) {
    hloader.key   = md2cmds_cmd_kvs[i].k;
    hloader.data  = md2cmds_cmd_kvs[i].v;
    err = hsearch_r( hloader, ENTER, &hrtnval, &md2cmds_hmap);
    if( err == 0) {
        lslogging_log_message( "md2cmds_init: hsearch_r
            returned an error for item %d: %s", i, strerror( errno));
    }
}

if( pgpmac_use_pg) {
    for( i=0; i<sizeof(md2cmds_cmd_pg_kvs)/sizeof(
        md2cmds_cmd_pg_kvs[0]); i++) {
        hloader.key   = md2cmds_cmd_pg_kvs[i].k;
        hloader.data  = md2cmds_cmd_pg_kvs[i].v;
        err = hsearch_r( hloader, ENTER, &hrtnval, &md2cmds_hmap);
        if( err == 0) {
            lslogging_log_message( "md2cmds_init: hsearch_r
                returned an error for item %d: %s", i, strerror( errno));
        }
    }
}
}

```

#### 7.9.3.14 int md2cmds\_is\_moving( )

returns non-zero if we think a motor is moving, 0 otherwise

Definition at line 199 of file md2cmds.c.

```

{
int rtn;

pthread_mutex_lock( &md2cmds_moving_mutex);
rtn = md2cmds_moving_count != 0;
pthread_mutex_unlock( &md2cmds_moving_mutex);

```

```
    return rtn;
}
```

### 7.9.3.15 void md2cmds\_kappaphi\_move ( double kappa\_deg, double phi\_deg )

Definition at line 1020 of file md2cmds.c.

```
{

int kc, pc;

// coordinate system 7
// 1 << (coord sys no - 1) = 64

kc = md2cmds_prep_axis( kappa, kappa_deg);
pc = md2cmds_prep_axis( kappa, phi_deg);

// ;150           LS-CAT Move X, Y Absolute
// ;                 Q20      = X Value
// ;                 Q21      = Y Value
// ;                 Q100     = 1 << (coord sys no - 1)

lspmac_SockSendDPLine( "kappaphi_move", "&7 Q20=%d
                           Q21=%d Q100=64", kc, pc);
}
```

### 7.9.3.16 void md2cmds\_maybe\_done\_homing\_cb ( char \* event )

Track motors homing.

Definition at line 996 of file md2cmds.c.

```
{

pthread_mutex_lock( &md2cmds_homing_mutex);

if( strstr( event, "Homing" ) != NULL) {
    if( md2cmds_homing_count == -1)
        md2cmds_homing_count = 1;
    else
        md2cmds_homing_count++;
} else {
    if( md2cmds_homing_count > 0)
        md2cmds_homing_count--;
}

if( md2cmds_homing_count != 0)
    lsredis_setstr( md2cmds_md_status_code,
                    "%s", "4");

if( md2cmds_homing_count == 0)
    pthread_cond_signal( &md2cmds_homing_cond);

pthread_mutex_unlock( &md2cmds_homing_mutex);
}
```

### 7.9.3.17 void md2cmds\_maybe\_done\_moving\_cb ( char \* event )

Track how many motors are moving.

Definition at line 966 of file md2cmds.c.

```
{

pthread_mutex_lock( &md2cmds_moving_mutex);
if( strstr( event, "Moving" ) != NULL) {
    //
    // -1 is a flag indicating we're expecting some action
    //
    if( md2cmds_moving_count == -1)
        md2cmds_moving_count = 1;
    else
        md2cmds_moving_count++;
```

```

} else {
//
//
if( md2cmds_moving_count > 0)
    md2cmds_moving_count--;
}

lsredis_setstr( md2cmds_md_status_code,
    "%s", md2cmds_moving_count ? "4" : "3");
lslogging_log_message( "md2cmds_maybe_done_moving_cb:
    status code %ld", lsredis_getl( md2cmds_md_status_code
))));

if( md2cmds_moving_count == 0)
    pthread_cond_signal( &md2cmds_moving_cond);
    pthread_mutex_unlock( &md2cmds_moving_mutex);

}

```

### 7.9.3.18 void md2cmds\_maybe\_rotate\_done\_cb ( char \* event )

Now that we are done with the 360 rotation lets rehome right quick.

Definition at line 1496 of file md2cmds.c.

```

{
if( rotating) {
    rotating = 0;
    lsevents_send_event( "Rotate Done");
}
}
```

### 7.9.3.19 void md2cmds\_move\_prep ( )

prepare for new movements

Definition at line 143 of file md2cmds.c.

```

{
pthread_mutex_lock( &md2cmds_moving_mutex);
lsredis_setstr( md2cmds_md_status_code,
    "%s", "4");
lslogging_log_message( "md2cmds_move_prep: status code
    %ld", lsredis_getl( md2cmds_md_status_code));
md2cmds_moving_count = md2cmds_moving_count
    ? md2cmds_moving_count : -1;
pthread_mutex_unlock( &md2cmds_moving_mutex);
}
```

### 7.9.3.20 int md2cmds\_move\_wait ( double timeout\_secs )

Wait for all the motions requested to complete.

#### Parameters

<i>timeout_secs</i>	Double value of seconds to wait
---------------------	---------------------------------

There are two waits involved: First to wait for the first "Moving" to be seen and second to wait for the last "In Position". The timeout specified here is the sum of the two.

returns 0 on success and 1 if we timedout.

Definition at line 162 of file md2cmds.c.

```

{
double isecs, fsecs;
struct timespec timeout, now;
```

```

int err;

clock_gettime( CLOCK_REALTIME, &now);

fsecs = modf( timeout_secs, &isecs);

timeout.tv_sec = now.tv_sec + (long)floor( isecs);
timeout.tv_nsec = now.tv_nsec + (long)floor( fsecs * 1.0e9);

timeout.tv_sec += timeout.tv_nsec / 1000000000;
timeout.tv_nsec %= 1000000000;

err = 0;
pthread_mutex_lock( &md2cmds_moving_mutex);
while( err == 0 && md2cmds_moving_count == -1)
    err = pthread_cond_timedwait( &md2cmds_moving_cond, &
        md2cmds_moving_mutex, &timeout);

if( err == ETIMEDOUT) {
    pthread_mutex_unlock( &md2cmds_moving_mutex);
    return 1;
}

err = 0;
while( err == 0 && md2cmds_moving_count > 0)
    err = pthread_cond_timedwait( &md2cmds_moving_cond, &
        md2cmds_moving_mutex, &timeout);
pthread_mutex_unlock( &md2cmds_moving_mutex);

if( err == ETIMEDOUT)
    return 1;
return 0;
}

```

### 7.9.3.21 int md2cmds\_moveAbs ( const char \* ccmd )

Move a motor to the position requested Returns non zero on error.

#### Parameters

in	ccmd	The full command string to parse, ie, "moveAbs omega 180"
----	------	---

Definition at line 466 of file md2cmds.c.

```

{
char *cmd;
char *ignore;
char *ptr;
char *mtr;
char *pos;
double fpos;
char *endptr;
lspmac_motor_t *mp;
int err;

// ignore nothing
if( ccmd == NULL || *ccmd == 0) {
    return 1;
}

// operate on a copy of the string since strtok_r will modify its argument
//
cmd = strdup( ccmd);

// Parse the command string
//
ignore = strtok_r( cmd, " ", &ptr);
if( ignore == NULL) {
    lslogging_log_message( "md2cmds_moveAbs: ignoring
        blank command '%s'", cmd);
    free( cmd);
    return 1;
}

// The first string should be "moveAbs" cause that's how we got here.
// Toss it.

mtr = strtok_r( NULL, " ", &ptr);
if( mtr == NULL) {

```

```

lslogging_log_message( "md2cmds_moveAbs: missing motor
    name");
free( cmd);
return 1;
}

mp = lspmac_find_motor_by_name( mtr);
if( mp == NULL) {
    lslogging_log_message( "md2cmds_moveAbs: cannot find
        motor %s", mtr);
    free( cmd);
    return 1;
}

pos = strtok_r( NULL, " ", &ptr);
if( pos == NULL) {
    lslogging_log_message( "md2cmds_moveAbs: missing
        position");
    free( cmd);
    return 1;
}

fpos = strtod( pos, &endptr);
if( pos == endptr) {
    //
    // Maybe we have a preset. Give it a whirl
    // In any case we are done here.
    //
    err = lspmac_move_preset_queue( mp, pos);
    free( cmd);
    return err;
}

if( mp != NULL && mp->moveAbs != NULL) {
    pgpmac_printf( "Moving %s to %f\n", mtr, fpos);
    err = mp->moveAbs( mp, fpos);
}

free( cmd);
return err;
}

```

### 7.9.3.22 int md2cmds\_moveRel( const char \* ccmd )

Move a motor to the position requested Returns non zero on error.

#### Parameters

in	ccmd	The full command string to parse, ie, "moveAbs omega 180"
----	------	---

Definition at line 545 of file md2cmds.c.

```

{
char *cmd;
char *ignore;
char *ptr;
char *mtr;
char *pos;
double fpos;
char *endptr;
lspmac_motor_t *mp;
int err;

// ignore nothing
if( ccmd == NULL || *ccmd == 0) {
    return 1;
}

// operate on a copy of the string since strtok_r will modify its argument
// cmd = strdup( ccmd);

// Parse the command string
//
ignore = strtok_r( cmd, " ", &ptr);
if( ignore == NULL) {
    lslogging_log_message( "md2cmds_moveAbs: ignoring
        blank command '%s'", cmd);
}

```

```

    free( cmd);
    return 1;
}

// The first string should be "moveAbs" cause that's how we got here.
// Toss it.

mtr = strtok_r( NULL, " ", &ptr);
if( mtr == NULL) {
    lslogging_log_message( "md2cmds_moveRel: missing motor
        name");
    free( cmd);
    return 1;
}

mp = lspmac_find_motor_by_name( mtr);

if( mp == NULL) {
    lslogging_log_message( "md2cmds_moveRel: cannot find
        motor %s", mtr);
    free( cmd);
    return 1;
}

pos = strtok_r( NULL, " ", &ptr);
if( pos == NULL) {
    lslogging_log_message( "md2cmds_moveRel: missing
        position");
    free( cmd);
    return 1;
}

fpos = strtod( pos, &endptr);
if( pos == endptr) {
    //
    // No incremental position found
    //
    lslogging_log_message( "md2cmds_moveRel: no new
        position requested");
    return 1;
}

if( mp != NULL && mp->moveAbs != NULL) {
    lslogging_log_message( "Moving %s by %f\n", mtr, fpos)
    ;
    err = mp->moveAbs( mp, lspmac_getPosition(mp) +
        fpos);
}
free( cmd);
return err;
}

```

### 7.9.3.23 void md2cmds\_mvcenter\_move ( double cx, double cy, double ax, double ay, double az )

Move the centering and alignment tables.

#### Parameters

in	<i>cx</i>	Requested Centering Table X
in	<i>cy</i>	Requested Centering Table Y
in	<i>ax</i>	Requested Alignment Table X
in	<i>ay</i>	Requested Alignment Table Y
in	<i>az</i>	Requested Alignment Table Z

Definition at line 939 of file md2cmds.c.

```

{
// centering stage is coordinate system 2
// alignment stage is coordinate system 3
//

double cx_cts, cy_cts, ax_cts, ay_cts, az_cts;
cx_cts = md2cmds_prep_axis( cenz,   cx);

```

```

    cy_cts = md2cmds_prep_axis( ceny,   cy);
    ax_cts = md2cmds_prep_axis( alignx, ax);
    ay_cts = md2cmds_prep_axis( aligny, ay);
    az_cts = md2cmds_prep_axis( alignz, az);

    lspmac_SockSendDPLine( NULL, "&2 Q100=2 Q20=%1f
                                Q21=%1f B150R", cx_cts, cy_cts);
    lspmac_SockSendDPLine( "mvcenter_move", "&3 Q100=4
                                Q30=%1f Q31=%1f Q32=%1f B160R", ax_cts, ay_cts, az_cts);
}
}

```

### 7.9.3.24 void md2cmds\_organs\_move\_presets ( char \* pay, char \* paz, char \* pcy, char \* pcz, char \* psz )

Definition at line 232 of file md2cmds.c.

```

{
double ay,     az,    cy,    cz,    sz;
int    cay,   caz,   ccy,   ccz,   csz;
int err;

err = lsredis_find_preset( apery->name, pay, &ay)
;
if( err == 0) {
    lslogging_log_message( "md2cmds_move_organs_presets:
        no preset '%s' for motor '%s'", pay, apery->name);
    return;
}

err = lsredis_find_preset( aperz->name, paz, &az)
;
if( err == 0) {
    lslogging_log_message( "md2cmds_move_organs_presets:
        no preset '%s' for motor '%s'", paz, aperz->name);
    return;
}

err = lsredis_find_preset( capy->name, pcy, &cy);
if( err == 0) {
    lslogging_log_message( "md2cmds_organs_move_presets:
        no preset '%s' for motor '%s'", pcy, capy->name);
    return;
}

err = lsredis_find_preset( capz->name, pcz, &cz);
if( err == 0) {
    lslogging_log_message( "md2cmds_organs_move_presets:
        no preset '%s' for motor '%s'", pcz, capz->name);
    return;
}

err = lsredis_find_preset( scint->name, psz, &sz)
;
if( err == 0) {
    lslogging_log_message( "md2cmds_organs_move_presets:
        no preset '%s' for motor '%s'", psz, scint->name);
    return;
}

cay = md2cmds_prep_axis( apery, ay);
caz = md2cmds_prep_axis( aperz, az);
ccy = md2cmds_prep_axis( capy, cy);
ccz = md2cmds_prep_axis( capz, cz);
csz = md2cmds_prep_axis( scint, sz);

//          LS-CAT Move U, V, W, X, Y, Z Absolute
//          Q40      = X Value
//          Q41      = Y Value
//          Q42      = Z Value
//          Q43      = U Value
//          Q44      = V Value
//          Q45      = W Value

lspmac_SockSendDPLine( "organs", "&5 Q40=0 Q41=%d Q42=%d
                            Q43=%d Q44=%d Q45=%d Q100=16 B170R", cay, caz, ccy, ccz, csz);
}
}

```

### 7.9.3.25 int md2cmds\_phase\_beamLocation( )

Go to beam location phase.

Definition at line 798 of file md2cmds.c.

```
{
double move_time;
int mmask, err;

lsevents_send_event( "Mode beamLocation Starting");

mmask = 0;
err = lspmac_est_move_time( &move_time, &mmask,
                           //motor jog, preset,           position if no preset

                           NULL,                 0.0,          kappa,      0,
                           omega,      0, NULL,      0.0,
                           apery,      0, "In",      0.0,
                           aperz,      0, "In",      0.0,
                           capy,      0, "In",      0.0,
                           capz,      0, "In",      0.0,
                           scint,      0, "Scintillator", 0.0,
                           blight,     1, NULL,      0.0,
                           blight_ud, 1, NULL,      0.0,
                           zoom,       0, NULL,      1.0,
                           cryo,       1, NULL,      0.0,
                           fluo,       1, NULL,      0.0,
                           NULL);

if( err) {
    lsevents_send_event( "Mode beamLocation Aborted");
    return err;
}

err = lspmac_est_move_time_wait( move_time + 2.0,
                                 mmask, blight_ud, cryo, fluo, NULL);
if( err) {
    lsevents_send_event( "Mode beamLocation Aborted");
    return err;
}

lsevents_send_event( "Mode beamLocation Done");
return 0;
}
```

### 7.9.3.26 int md2cmds\_phase\_center( )

Go to center phase.

Definition at line 723 of file md2cmds.c.

```
{
double move_time;
int mmask, err;

lsevents_send_event( "Mode center Starting");
//                                     // Move 'em

//                                     //

mmask = 0;
err = lspmac_est_move_time( &move_time, &mmask,
                           omega,      0, NULL,      0.0,
                           kappa,      0, NULL,      0.0,
                           phi,        0, NULL,      0.0,
                           apery,      0, "In",      0.0,
                           aperz,      0, "In",      0.0,
                           capy,      0, "In",      0.0,
                           capz,      0, "In",      0.0,
                           scint,      0, "Cover",   0.0,
                           blight_ud, 1, NULL,      1.0,
```

```

        zoom,      0, NULL,    1.0,
        cryo,     1, NULL,    0.0,
        fluo,     1, NULL,    0.0,
        NULL);

if( err) {
lsevents_send_event( "Mode center Aborted");
return err;
}

err = lspmac_est_move_time_wait( move_time + 2.0,
mmask, cryo, fluo, NULL);
if( err) {
lsevents_send_event( "Mode center Aborted");
return err;
}

lsevents_send_event( "Mode center Done");
return 0;
}

```

### 7.9.3.27 int md2cmds\_phase\_change ( const char \* ccmd )

Move md2 devices to a preconfigured state.

- EMBL calls these states "phases" and this language is partially retained here \*\*

#### Parameters

<i>ccmd</i>	The full text of the command that sent us here
-------------	--

Definition at line 879 of file md2cmds.c.

```

{
char *cmd;
char *ignore;
char *ptr;
char *mode;
int err;

if( ccmd == NULL || *ccmd == 0)
return 1;

// use a copy as strtok_r modifies the string it is parsing

//


cmd = strdup( ccmd);

ignore = strtok_r( cmd, " ", &ptr);
if( ignore == NULL) {
lslogging_log_message( "md2cmds_phase_change: ignoring
empty command string (how did we let things get this far?");
free( cmd);
return 1;
}

//

// ignore should point to "mode" cause that's
how we got here. Ignore it

//


mode = strtok_r( NULL, " ", &ptr);
if( mode == NULL) {
lslogging_log_message( "md2cmds_phase_change: no mode
specified");
return 1;
}

if( md2cmds_is_moving()) {
int err;

```

```

lspmac_SockSendDPControlChar( "Aborting Motions
", '\x01');
err = md2cmds_move_wait( 2.0);
if( err) {
    lslogging_log_message( "md2cmds_phase_change: Timed
        out waiting for previous moves to finish");
    return 1;
}
}

//



                // Tangled web. Probably not worth fixing.
O(N) but N is 6.

//



if( strcmp( mode, "manualMount") == 0) {
    err = md2cmds_phase_manualMount();
} else if( strcmp( mode, "robotMount") == 0) {
    err = md2cmds_phase_robotMount();
} else if( strcmp( mode, "center") == 0) {
    err = md2cmds_phase_center();
} else if( strcmp( mode, "dataCollection") == 0) {
    err = md2cmds_phase_dataCollection();
} else if( strcmp( mode, "beamLocation") == 0) {
    err = md2cmds_phase_beamLocation();
} else if( strcmp( mode, "safe") == 0) {
    err = md2cmds_phase_safe();
}

free( cmd);
return err;
}

```

### 7.9.3.28 int md2cmds\_phase\_dataCollection( )

Go to data collection phase.

Definition at line 762 of file md2cmds.c.

```

{
double move_time;
int mmask, err;

lsevents_send_event( "Mode dataCollection Starting");

mmask = 0;
err = lspmac_est_move_time( &move_time, &mmask,
                            apery,      1, "In",     0.0,
                            aperz,      1, "In",     0.0,
                            capy,       1, "In",     0.0,
                            capz,       1, "In",     0.0,
                            scint,      1, "Cover",  0.0,
                            blight,     1, NULL,    0.0,
                            blight_ud, 1, NULL,    0.0,
                            cryo,       1, NULL,    0.0,
                            fluo,       1, NULL,    0.0,
                            NULL);

if( err) {
    lsevents_send_event( "Mode dataCollection Aborted");
    return err;
}

err = lspmac_est_move_time_wait( move_time + 2.0,
                                mmask, apery, aperz, capy, capz, scint, blight_ud,
                                cryo, fluo, NULL);
if( err) {
    lsevents_send_event( "Mode dataCollection Aborted");
    return err;
}

lsevents_send_event( "Mode dataCollection Done");
return 0;
}

```

## 7.9.3.29 int md2cmds\_phase\_manualMount( )

Go to the manual mount phase.

Definition at line 625 of file md2cmds.c.

```
{
double move_time;
int mmask, err;

lsevents_send_event( "Mode manualMount Starting");
// Move stuff
//
mmask = 0;
err = lspmac_est_move_time( &move_time, &mmask,
                           kappa,      0, "manualMount", 0.0,
                           omega,      0, "manualMount", 0.0,
                           phi,        0, NULL,          0.0,
                           aperz,      1, "Cover",       0.0,
                           capz,      1, "Cover",       0.0,
                           scint,      1, "Cover",       0.0,
                           blight,     1, NULL,          0.0,
                           blight_ud, 1, NULL,          0.0,
                           cryo,       1, NULL,          0.0,
                           fluo,       1, NULL,          0.0,
                           zoom,       0, NULL,          1.0,
                           NULL);

if( err) {
    lsevents_send_event( "Mode manualMount Aborted");
    return err;
}

// Wait for motion programs

//

err = lspmac_est_move_time_wait( move_time+2.0,
                                 mmask, aperz, scint, blight_ud, cryo, fluo, NULL);
if( err) {
    lsevents_send_event( "Mode manualMount Aborted");
    return err;
}

lsevents_send_event( "Mode manualMount Done");
return 0;
}
```

## 7.9.3.30 int md2cmds\_phase\_robotMount( )

Go to robot mount phase.

Definition at line 669 of file md2cmds.c.

```
{
double move_time;
int mmask, err;

lsevents_send_event( "Mode robotMount Starting");
md2cmds_home_prep();
//

// Move 'em

//
```

```

        lspmac_home1_queue( kappa);
lspmac_home1_queue( omega);
lspmac_home1_queue( kappa);

mmask = 0;
err = lspmac_est_move_time( &move_time, &mmask,
                            apery,      1, "In",      0.0,
                            aperz,      1, "In",      0.0,
                            capz,       1, "Cover",   0.0,
                            scint,      1, "Cover",   0.0,
                            blight,     1, NULL,     0.0,
                            blight_ud, 1, NULL,     0.0,
                            cryo,       1, NULL,     0.0,
                            fluo,       1, NULL,     0.0,
                            zoom,       0, NULL,     1.0,
                            NULL);

err = lspmac_est_move_time_wait( move_time + 2.0,
                                 mmask, apery, aperz, capz, scint, blight_ud, cryo,
                                 fluo, NULL);
if( err) {
    lsevents_send_event( "Mode robotMount Aborted");
    return err;
}

err = md2cmds_home_wait( 60.0);
if( err) {
    lslogging_log_message( "md2cmds_phase_robotMount:
                           timed out homing omega or kappa");
    lsevents_send_event( "Mode robotMount Aborted");
    return err;
}

md2cmds_home_prep();
lspmac_home1_queue( phi);
err = md2cmds_home_wait( 60.0);
if( err) {
    lslogging_log_message( "md2cmds_phase_robotMount:
                           timed out homing phi");
    lsevents_send_event( "Mode robotMount Aborted");
    return err;
}

lsevents_send_event( "Mode robotMount Done");
return 0;
}

```

### 7.9.3.31 int md2cmds\_phase\_safe( )

Go to safe phase.

Definition at line 837 of file md2cmds.c.

```

{
double move_time;
int mmask, err;

lsevents_send_event( "Mode safe Starting");

mmask = 0;
err = lspmac_est_move_time( &move_time, &mmask,
                            //motor    jog, preset,           position if no preset

                            kappa,      0, NULL,      0.0,
                            omega,      0, NULL,      0.0,
                            apery,      1, "In",      0.0,
                            aperz,      1, "Cover",   0.0,
                            capy,       1, "In",      0.0,
                            capz,       1, "Cover",   0.0,
                            scint,      1, "Cover",   0.0,
                            blight,     1, NULL,     0.0,
                            blight_ud, 1, NULL,     0.0,
                            zoom,       0, NULL,     1.0,
                            cryo,       1, NULL,     0.0,
                            fluo,       1, NULL,     0.0,
                            NULL);

```

```

if( err) {
    lsevents_send_event( "Mode safe Aborted");
    return err;
}

err = lsmpmac_est_move_time_wait( move_time + 2.0,
    mmask, apery, aperz, capy, capz, scint, blight_ud,
    cryo, fluo, NULL);
if( err) {
    lsevents_send_event( "Mode safe Aborted");
    return err;
}

lsevents_send_event( "Mode safe Done");
return 0;
}

```

### 7.9.3.32 double md2cmds\_prep\_axis ( lsmpmac\_motor\_t \* mp, double pos )

Definition at line 210 of file md2cmds.c.

```

{
double rtn;
double u2c;
double neutral_pos;

pthread_mutex_lock( &(mp->mutex));
u2c      = lsredis_getd( mp->u2c);
neutral_pos = lsredis_getd( mp->neutral_pos);

mp->motion_seen = 0;
mp->not_done     = 1;

rtn = u2c * (pos + neutral_pos);

pthread_mutex_unlock( &(mp->mutex));
return rtn;
}

```

### 7.9.3.33 void md2cmds\_push\_queue ( char \* action )

Definition at line 79 of file md2cmds.c.

```

{
if( pthread_mutex_trylock( &md2cmds_mutex) == 0) {
    strncpy( md2cmds_cmd, action, MD2CMDS_CMD_LENGTH
        -1);
    md2cmds_cmd[MD2CMDS_CMD_LENGTH-1] = 0;
    pthread_cond_signal( &md2cmds_cond);
    pthread_mutex_unlock( &md2cmds_mutex);
} else {
    lslogging_log_message( "md2cmds_push_queue: MD2
        command '%s' ignored. Already running '%s'", action, md2cmds_cmd);
}
}

```

### 7.9.3.34 int md2cmds\_rotate ( const char \* dummy )

Spin 360 and make a video (recenter first, maybe)

#### Parameters

<i>dummy</i>	Unused returns non-zero on error
--------------	----------------------------------

Definition at line 1331 of file md2cmds.c.

```

    {
double cx, cy, ax, ay, az, zm;
double bax, bay, baz;
int mmask;
int err;
double move_time;

mmask = 0;
// BLUMax disables scintilator here.
//

// get the new center information
//
lspg_getcenter_call();
lspg_getcenter_wait();

// put up the back light
blight_ud->moveAbs( blight_ud, 1);

//
// Get ready to move our motors
md2cmds_home_prep();

//
// make sure omega is homed
//
lspmac_home1_queue( omega);
//
// Grab the current positions
//
cx = lspmacGetPosition( cenx);
cy = lspmacGetPosition( ceny);
ax = lspmacGetPosition( alignx);
ay = lspmacGetPosition( aligny);
az = lspmacGetPosition( alignz);

lslogging_log_message( "md2cmds_rotate: actual positions
    cx %f, cy %f, ax %f, ay %f, az %f", cx, cy, ax, ay, az);

if( lspg_getcenter.no_rows_returned) {
//
// Always specify zoom even if no other center information is found
//
zm = 1;
} else {
lslogging_log_message( "md2cmds_rotate: getcenter
    returned dcx %f, dcy %f, dax %f, day %f, daz %f, zoom %d",
    lspg_getcenter.dcx, lspg_getcenter
    .dcy, lspg_getcenter.dax, lspg_getcenter.day
    , lspg_getcenter.daz,lspg_getcenter.zoom);

if( lspg_getcenter.zoom_isnull == 0) {
    zm = lspg_getcenter.zoom;
} else {
    zm = 1.0;
}

if( lspg_getcenter.dcx_isnull == 0)
    cx += lspg_getcenter.dcx;

if( lspg_getcenter.dcy_isnull == 0)
    cy += lspg_getcenter.dcy;

//
// Slightly complicated procedure for alignment stage since we might want
// to update
// the presets. Use the preset Back_Vector to calculate the new Back
// preset from our
// current position.
//
if( lspg_getcenter.dax_isnull == 0) {
    err = lsredis_find_preset( "align.x", "Back_Vector", &
    bax);
    if( err == 0)
        bax = 0.0;
    bax += lspg_getcenter.dax;
    lsredis_set_preset( "align.x", "Back", bax);

    ax += lspg_getcenter.dax;
    lsredis_set_preset( "align.x", "Beam", ax);
}

if( lspg_getcenter.day_isnull == 0) {

```

```

err = lsredis_find_preset( "align.y", "Back_Vector", &
    bay);
if( err == 0)
    bay = 0.0;
bay += lsgc_getcenter.day;
lsredis_set_preset( "align.y", "Back", bay);

ay += lsgc_getcenter.day;
lsredis_set_preset( "align.y", "Beam", ay);
}

if( lsgc_getcenter.daz_isnull == 0) {
    err = lsredis_find_preset( "align.z", "Back_Vector", &
        baz);
    if( err == 0)
        baz = 0.0;
    baz += lsgc_getcenter.daz;
    lsredis_set_preset( "align.z", "Back", baz);

    az += lsgc_getcenter.daz;
    lsredis_set_preset( "align.z", "Beam", az);
}
lsgc_getcenter_done();

if( lspmac_est_move_time( &move_time, &mmask,
    scint, 1, "Cover", 0.0,
    capz, 1, "Cover", 0.0,
    cenx, 0, NULL, cx,
    ceny, 0, NULL, cy,
    alignx, 0, NULL, ax,
    aligny, 0, NULL, ay,
    alignz, 0, NULL, az,
    zoom, 1, NULL, zm,
    NULL) ) {
    lslogging_log_message( "md2cmds_rotate: organ motion
        request failed");
    lsevents_send_event( "Rotate Aborted");
    return 1;
}

if( lspmac_est_move_time_wait( move_time + 2.0,
    mmask, scint, capz, zoom, NULL) ) {
    lslogging_log_message( "md2cmds_rotate: organ motion
        timed out %f seconds", move_time + 2.0);
    lsevents_send_event( "Rotate Aborted");
    return 1;
}

if( md2cmds_home_wait( 20.0)) {
    lslogging_log_message( "md2cmds_rotate: homing motors
        timed out. Rotate aborted");
    lsevents_send_event( "Rotate Aborted");
    return 1;
}

// Report new center positions
cx = lspmac_getPosition( cenx);
cy = lspmac_getPosition( ceny);
ax = lspmac_getPosition( alignx);
ay = lspmac_getPosition( aligny);
az = lspmac_getPosition( alignz);
lsgc_query_push( NULL, "SELECT px.applycenter( %.3f, %.3f,
    %.3f, %.3f, %.3f, %.3f)", cx, cy, ax, ay, az, lspmac_getPosition
    (kappa), lspmac_getPosition( phi));

lslogging_log_message( "md2cmds_rotate: done with
    applycenter");
lspmac_video_rotate( 4.0);
lslogging_log_message( "md2cmds_rotate: starting
    rotation");
rotating = 1;

return 0;
}

```

### 7.9.3.35 void md2cmds\_rotate\_cb ( char \* event )

Tell the database about the time we went through omega=zero.

This should trigger the video feed server to starting making a movie.

Definition at line 1481 of file md2cmds.c.

```

    {
struct tm t;
int usecs;

localtime_r( &(omega_zero_time.tv_sec), &t);

usecs = omega_zero_time.tv_nsec / 1000;
lspg_query_push( NULL, "SELECT px.trigcam('%d-%d-%d
%d:%d:%d.%06d', %d, 0.0, 90.0)",
                  t.tm_year+1900, t.tm_mon+1, t.tm_mday, t.tm_hour, t.tm_min,
t.tm_sec, usecs,
                  (int)(lspmac_getPosition( zoom)));
}

```

### 7.9.3.36 void md2cmds\_run( )

Start up the thread.

Definition at line 1952 of file md2cmds.c.

```

{
pthread_create( &md2cmds_thread, NULL,
                md2cmds_worker, NULL);
lsevents_add_listener( "^omega crossed zero$",
    md2cmds_rotate_cb);
lsevents_add_listener( "^omega In Position$",
    md2cmds_maybe_rotate_done_cb);
lsevents_add_listener( ".+ (Moving|In Position)$",
    md2cmds_maybe_done_moving_cb);
lsevents_add_listener( "(.+) (Homing|Homed)$",
    md2cmds_maybe_done_homing_cb);
lsevents_add_listener( "^capz (Moving|In Position)$",
    md2cmds_time_capz_cb);
lsevents_add_listener( "^Coordsys 1 Stopped$",
    md2cmds_coordsys_1_stopped_cb);
lsevents_add_listener( "^Coordsys 2 Stopped$",
    md2cmds_coordsys_2_stopped_cb);
lsevents_add_listener( "^Coordsys 3 Stopped$",
    md2cmds_coordsys_3_stopped_cb);
lsevents_add_listener( "^Coordsys 4 Stopped$",
    md2cmds_coordsys_4_stopped_cb);
lsevents_add_listener( "^Coordsys 5 Stopped$",
    md2cmds_coordsys_5_stopped_cb);
lsevents_add_listener( "^Coordsys 7 Stopped$",
    md2cmds_coordsys_7_stopped_cb);
lsevents_add_listener( "^cam.zoom Moving$",
    md2cmds_set_scale_cb);
}

```

### 7.9.3.37 int md2cmds\_run\_cmd( const char \* cmd )

Definition at line 1646 of file md2cmds.c.

```

{
int err, i;
lspmac_motor_t *mp;
regmatch_t pmatch[16];
char cp[64];

if( strlen(cmd) > sizeof( cp)-1) {
    lslogging_log_message( "md2cmds_run_cmd: command too
        long '%s'", cmd);
    return 1;
}

err = regexec( &md2cmds_cmd_regex, cmd, 16, pmatch, 0);
if( err) {
    lslogging_log_message( "md2cmds_run_cmd: no match
        found from '%s'", cmd);
    return 1;
}

for( i=0; i<16; i++) {

```

```

    if( pmatch[i].rm_so == -1)
        continue;
    lslogging_log_message( "md2cmds_run_cmd: %d '%.*s'", i
        , pmatch[i].rm_eo - pmatch[i].rm_so, cmd+pmatch[i].rm_so);
}

// 
// get motor name
//
snprintf( cp, sizeof( cp)-1, "%.*s", pmatch[4].rm_eo - pmatch[4].rm_so, cmd+
    pmatch[4].rm_so);
cp[sizeof( cp)-1] = 0;

mp = lspmac_find_motor_by_name( cp);
if( mp == NULL) {
    lslogging_log_message( "md2cmds_run_cmd: could not
        find motor '%s'", cp);
    return 1;
}

if( pmatch[5].rm_so != -1) {
    if( strncmp( cmd+pmatch[5].rm_so, "home", pmatch[5].rm_eo-pmatch[5].rm_so)
        ==0) {
        lslogging_log_message( "md2cmds_run_cmd: homing
            motor '%s'", cp);
        lspmac_home1_queue( mp);
    } else if( strncmp( cmd+pmatch[5].rm_so, "stop", pmatch[5].rm_eo-pmatch[5].
        rm_so)==0) {
        lslogging_log_message( "md2cmds_run_cmd: stoping
            motor '%s'", cp);
        lspmac_abort();
    }
}

return 0;
}

```

### 7.9.3.38 int md2cmds\_set( const char \* cmd )

Definition at line 1747 of file md2cmds.c.

```

{
int err;
lsredis_obj_t *p;
lspmac_motor_t *mp;
regmatch_t pmatch[16];
char cp[64];
char *rp;

if( strlen(cmd) > sizeof( cp)-1) {
    lslogging_log_message( "md2cmds_set: command too long
        '%s'", cmd);
    return 1;
}

lslogging_log_message( "md2cmds_set: received '%s'", cmd
    );

err = regexec( &md2cmds_cmd_regex, cmd, 16, pmatch, 0);
if( err) {
    lslogging_log_message( "md2cmds_set: no match found
        from '%s'", cmd);
    return 1;
}

if( pmatch[2].rm_so == -1) {
    lslogging_log_message( "md2cmds_set: could not parse
        preset name from '%s'", cmd);
    return 1;
}

// 
// get motor name
//
snprintf( cp, sizeof( cp)-1, "%.*s", pmatch[3].rm_eo - pmatch[3].rm_so, cmd+
    pmatch[3].rm_so);
cp[sizeof( cp)-1] = 0;

mp = lspmac_find_motor_by_name( cp);

```

```

if( mp == NULL) {
    lslogging_log_message( "md2cmds_set: could not find
        motor '%s'", cp);
    return 1;
}

// 
// get redis preset position name
//

p = lsredis_get_obj( "%.*s.position", pmatch[2].rm_eo - pmatch
    [2].rm_so, cmd+pmatch[2].rm_so);
if( p == NULL) {
    lslogging_log_message( "md2cmds_set: could not find
        preset name in '%s'", cmd);
    return 1;
}

rp = lsredis_getstr( mp->redis_position);

//
// set the preset to the current position
//
lsredis_setstr( p, "%s", rp);
lsevents_send_event( "Preset Changed %s", p->events_name
    );
free( rp);
return 0;
}

```

#### 7.9.3.39 void md2cmds\_set\_scale\_cb ( char \* event )

Fix up xscale and yscale when zoom changes xscale and yscale have units of microns per pixel.

Definition at line 1508 of file md2cmds.c.

```

{
int mag;
lsredis_obj_t *p1, *p2;
char *vp;

pthread_mutex_lock( &zoom->mutex);
mag = zoom->requested_position;
pthread_mutex_unlock( &zoom->mutex);

p1 = lsredis_get_obj( "cam.xScale");
p2 = lsredis_get_obj( "cam.zoom.%d.ScaleX", mag);

vp = lsredis_getstr( p2);
lsredis_setstr( p1, vp);
free( vp);

p1 = lsredis_get_obj( "cam.yScale");
p2 = lsredis_get_obj( "cam.zoom.%d.ScaleY", mag);

vp = lsredis_getstr( p2);
lsredis_setstr( p1, vp);
free( vp);
}

```

#### 7.9.3.40 int md2cmds\_settransferpoint ( const char \* cmd )

Definition at line 1695 of file md2cmds.c.

```

{
double ax, ay, az, cx, cy;

md2cmds_home_prep();

//
// Home Kappa
//
lspmac_home1_queue( kappa);

//
// Home omega

```

```

// lspmac_home1_queue( omega);

if( md2cmds_home_wait( 30.0) {
    lslogging_log_message( "md2cmds_settransferpoint:
        homing routines taking too long. Aborting transfer.");
    lsevents_send_event( "Settransferpoint Aborted");
    return 1;
}

md2cmds_home_prep();
// Home phi (whatever that means)
// lspmac_home1_queue( phi);

// Wait for the homing routines to finish
//
if( md2cmds_home_wait( 30.0) {
    lslogging_log_message( "md2cmds_settransferpoint:
        homing routines taking too long. Aborting transfer.");
    lsevents_send_event( "Settransferpoint Aborted");
    return 1;
}

// get positions we'll be needed to report to postgres
//
ax = lspmac_getPosition(alignx);
ay = lspmac_getPosition(aligny);
az = lspmac_getPosition(alignz);
cx = lspmac_getPosition(cenx);
cy = lspmac_getPosition(ceny);

lspg_query_push( NULL, "SELECT px.settransferpoint( %0.3f,
    %0.3f, %0.3f, %0.3f)", ax, ay, az, cx, cy);

lsevents_send_event( "Settransferpoint Done");
return 0;
}

```

#### 7.9.3.41 int md2cmds\_test ( const char \* *dummy* )

Run the test routine(s)

##### Parameters

<i>dummy</i>	Unused
--------------	--------

Definition at line 1640 of file md2cmds.c.

```

{
lctest_main();
return 0;
}

```

#### 7.9.3.42 void md2cmds\_time\_capz\_cb ( char \* *event* )

Time the capillary motion for the transfer routine.

< track the time spent moving capz

Definition at line 1534 of file md2cmds.c.

```

{
static struct timespec capz_timestarted;
struct timespec now;
int nsec, sec;

if( strstr( event, "Moving") != NULL) {
    clock_gettime( CLOCK_REALTIME, &capz_timestarted);
} else {
    clock_gettime( CLOCK_REALTIME, &now);
}

```

```
    sec = now.tv_sec - capz_timestarted.tv_sec;
    nsec = 0;
    if( now.tv_nsec > capz_timestarted.tv_nsec) {
        sec--;
        nsec += 1000000000;
    }
    nsec += now.tv_nsec - capz_timestarted.tv_nsec;
    md2cmds_capz_moving_time = sec + nsec / 1000000000.
}
}
```

#### 7.9.3.43 int md2cmds\_transfer ( const char \* *dummy* )

Transfer a sample.

## Parameters

*dummy* Unused

Definition at line 290 of file md2cmds.c.

```

    }

int nextsample, abort_now;
double ax, ay, az, cx, cy, horz, vert, oref;
int err;
int mmask;
double move_time;

nextsample = lspg_nextsample_all( &err);
if( err) {
    lslogging_log_message( "md2cmds_transfer: no sample
        requested to be transferred, false alarm");
    return 1;
}

//  

// BLUMax sets up an abort dialogbox here. Probably we should figure out how
// we are going to handle that.
//  

//  

// Wait for motors to stop
//  

if( md2cmds_is_moving()) {
    lslogging_log_message( "md2cmds_transfer: Waiting for
        previous motion to finish");
    if( md2cmds_move_wait( 30.0) {
        lslogging_log_message( "md2cmds_transfer: Timed out
            waiting for previous motion to finish. Aborting transfer");
    }
}

//  

// get positions we'll be needed to report to postgres
//  

ax = lspmac_getPosition(alignx);
ay = lspmac_getPosition(aligny);
az = lspmac_getPosition(alignz);
cx = lspmac_getPosition(cenx);
cy = lspmac_getPosition(ceny);
oref = lsredis_getd(lsredis_get_obj( "
    omega.reference")) * M_PI/180.;

horz = cx * cos(oref) + cy * sin(oref);
vert = cx * sin(oref) - cy * cos(oref);

mmask = 0;
err = lspmac_est_move_time( &move_time, &mmask,
    apery,      1, "In",      0.0,
    aperz,      1, "Cover",   0.0,
    capy,       1, "In",      0.0,
    capz,       1, "Cover",   0.0,
    scint,      1, "Cover",   0.0,
    blight_ud, 1, NULL,     0.0,
    fluo,       1, NULL,     0.0,
    NUL,L);
}

```

```

lspg_starttransfer_call( nextsample,
    lspmact_getBIPosition( sample_detected ), ax,
    ay, az, horz, vert, move_time);

md2cmds_home_prep();

// Home Kappa
// lspmact_home1_queue( kappa );

// Home omega
// lspmact_home1_queue( omega );

// Wait for the kappa/omega homing routines to finish
//
if( md2cmds_home_wait( 30.0 ) {
    lslogging_log_message( "md2cmds_transfer: kappa/omega
        homing routines taking too long. Aborting transfer." );
    lsevents_send_event( "Transfer Aborted" );
    return 1;
}

// Home phi (whatever that means)
//
md2cmds_home_prep();
lspmact_home1_queue( phi );

// Now let's get back to postgresql (remember our query so long ago?)
//
lspg_starttransfer_wait();

//
// It's possible that the sample that's mounted is unknown to the robot.
// If so then we need to abort after we're done moving stuff
//
lslogging_log_message( "md2cmds_transfer:
    no_rows_returned %d, starttransfer %d",
    lspg_starttransfer.no_rows_returned
    , lspg_starttransfer.starttransfer );
if( lspg_starttransfer.no_rows_returned ||
    lspg_starttransfer.starttransfer != 1 )
    abort_now = 1;
else
    abort_now = 0;

lspg_starttransfer_done();

//
// Wait for the homing routines to finish
//
if( md2cmds_home_wait( 30.0 ) {
    lslogging_log_message( "md2cmds_transfer: phi homing
        routine taking too long. Aborting transfer." );
    lsevents_send_event( "Transfer Aborted" );
    return 1;
}

//
// Wait for all those other motors to stop moving
//
err = lspmact_est_move_time_wait( move_time + 2.0,
    mmask, apery, aperz, capy, capz, scint, blight_ud,
    fluo, NULL );
if( err ) {
    lsevents_send_event( "Transfer Aborted" );
    return 1;
}

// TODO: check that all the motors are where we told them to go
//

// see if we have a sample mounted problem (is abort_now misnamed?)
//
if( abort_now ) {
    lslogging_log_message( "md2cmds_transfer: Apparently
        there is a sample mounted already but we don't know where it is supposed to go" );
    lsevents_send_event( "Transfer Aborted" );
    return 1;
}

```

```

}

// refuse to go on if we do not have positive confirmation that the backlight
// is down and the
// fluorescence detector is back (TODO: how about all those organs?)
//  

if( lspmac_getBIPosition( blight_down ) != 1 ||
    lspmac_getBIPosition( fluor_back ) != 1 ) {
    lslogging_log_message( "md2cmds_transfer: It looks
        like either the back light is not down or the fluorescence dectector is not back");
    lsevents_send_event( "Transfer Aborted");
    return 1;
}

//  

// Wait for the robot to unlock the cryo which signals us that we need to
// move the cryo back and drop air rights
//  

lspg_waitcryo_all();

// Move the cryo back
//  

cryo->moveAbs( cryo, 1);
lspmac_moveabs_wait( cryo, 10.0);

// simplest query yet!
lspg_query_push( NULL, "SELECT px.dropairrights()");

// wait for the result
// TODO: find an easy way out of this in case of error
//  

lspg_getcurrentsampleid_wait_for_id(
    nextsample);

// grab the airrights again
//  

lspg_demandairrights_all();

// Return the cryo
//  

cryo->moveAbs( cryo, 0);
lspmac_moveabs_wait( cryo, 10.0);

lsevents_send_event( "Transfer Done");

return 0;
}

```

#### 7.9.3.44 void\* md2cmds\_worker ( void \* *dummy* )

Our worker thread.

##### Parameters

<i>dummy</i>	[in] Unused but required by protocol
--------------	--------------------------------------

Definition at line 1812 of file md2cmds.c.

```

{
ENTRY hsearcher, *hrtnval;
char theCmd[32], *sp;
int i, err;
md2cmds_cmd_kv_t *cmdp;

pthread_mutex_lock( &md2cmds_mutex );

while( 1 ) {
    //
    // wait for someone to give us a command (and tell us they did so)
    //
    while( md2cmds_cmd[0] == 0 )
        pthread_cond_wait( &md2cmds_cond, &md2cmds_mutex
    );

```

```

// pull out the command name itself from the string we were given
//
for( i=0, sp=md2cmds_cmd; i<sizeof( theCmd)-1; i++, sp++) {
    if( *sp == 0 || *sp == ' ' ) {
        theCmd[i] = 0;
        break;
    }
    theCmd[i] = *sp;
}
theCmd[sizeof(theCmd)-1]=0;

hsearcher.key = theCmd;
hsearcher.data = NULL;

errno = 0;
err = hsearch_r( hsearcher, FIND, &hrtnval, &md2cmds_hmap );
if( err == 0 ) {
    lslogging_log_message( "md2cmds_worker: hsearch_r
        failed. theCmd = '%s' from string '%s'", theCmd, md2cmds_cmd );
    md2cmds_cmd[0] = 0;
    continue;
}
lslogging_log_message( "md2cmds_worker: Found command
    '%s', theCmd);
if( hrtnval != NULL) {
    cmdp = (md2cmds_cmd_kv_t *)hrtnval;
    err = cmdp->v( md2cmds_cmd );
    if( err ) {
        lslogging_log_message( "md2cmds_worker: Command
            failed: '%s', md2cmds_cmd );
        //
        // At this point we'd clear the queue but the queue is currently too
        short to bother doing that
        //
    }
}
md2cmds_cmd[0] = 0;
}
}

```

## 7.9.4 Variable Documentation

### 7.9.4.1 double md2cmds\_capz\_moving\_time = NAN [static]

Definition at line 32 of file md2cmds.c.

### 7.9.4.2 char md2cmds\_cmd[MD2CMDS\_CMD\_LENGTH]

our command;

Definition at line 24 of file md2cmds.c.

### 7.9.4.3 md2cmds\_cmd\_kv\_t md2cmds\_cmd\_kvs[] [static]

#### Initial value:

```

= {
    { "abort",           md2cmds_abort },
    { "changeMode",      md2cmds_phase_change },
    { "moveAbs",          md2cmds_moveAbs },
    { "moveRel",          md2cmds_moveRel },
    { "run",              md2cmds_run_cmd },
    { "test",             md2cmds_test },
    { "set",              md2cmds_set },
}

```

Definition at line 59 of file md2cmds.c.

#### 7.9.4.4 `md2cmds_cmd_kv_t md2cmds_cmd_pg_kvs[] [static]`

**Initial value:**

```
= {
  { "collect",           md2cmds_collect},
  { "rotate",            md2cmds_rotate},
  { "settransferpoint", md2cmds_settransferpoint},
  { "transfer",          md2cmds_transfer}
}
```

Definition at line 72 of file md2cmds.c.

#### 7.9.4.5 `regex_t md2cmds_cmd_regex [static]`

Definition at line 36 of file md2cmds.c.

#### 7.9.4.6 `pthread_cond_t md2cmds_cond`

condition to signal when it's time to run an md2 command

Definition at line 10 of file md2cmds.c.

#### 7.9.4.7 `struct hsearch_data md2cmds_hmap [static]`

Definition at line 34 of file md2cmds.c.

#### 7.9.4.8 `pthread_cond_t md2cmds_homing_cond`

coordinate homing and homed

Definition at line 18 of file md2cmds.c.

#### 7.9.4.9 `int md2cmds_homing_count = 0`

We've asked a motor to home.

Definition at line 17 of file md2cmds.c.

#### 7.9.4.10 `pthread_mutex_t md2cmds_homing_mutex`

our mutex;

Definition at line 19 of file md2cmds.c.

#### 7.9.4.11 `Isredis_obj_t* md2cmds_md_status_code`

Definition at line 26 of file md2cmds.c.

#### 7.9.4.12 `pthread_cond_t md2cmds_moving_cond`

wait for command to have been dequeued and run

coordinate call and response

Definition at line 14 of file md2cmds.c.

7.9.4.13 int md2cmds\_moving\_count = 0

Definition at line 22 of file md2cmds.c.

7.9.4.14 pthread\_mutex\_t md2cmds\_moving\_mutex

message passing between md2cmds and pg

Definition at line 15 of file md2cmds.c.

7.9.4.15 int md2cmds\_moving\_queue\_wait = 0

Definition at line 13 of file md2cmds.c.

7.9.4.16 pthread\_mutex\_t md2cmds\_mutex

mutex for the condition

Definition at line 11 of file md2cmds.c.

7.9.4.17 pthread\_t md2cmds\_thread [static]

Definition at line 28 of file md2cmds.c.

7.9.4.18 int rotating = 0 [static]

flag: when omega is in position after a rotate we want to re-home omega

Definition at line 30 of file md2cmds.c.

## 7.10 mk\_pgpmac\_redis.py File Reference

### Namespaces

- namespace [mk\\_pgpmac\\_redis](#)

### Functions

- def [mk\\_pgpmac\\_redis.mk\\_home](#)
- def [mk\\_pgpmac\\_redis.mk\\_active\\_init](#)
- def [mk\\_pgpmac\\_redis.mk\\_inactive\\_init](#)
- def [mk\\_pgpmac\\_redis.active\\_simulation](#)
- def [mk\\_pgpmac\\_redis.asis](#)

### Variables

- list [mk\\_pgpmac\\_redis.head](#) sys.argv[1]
- list [mk\\_pgpmac\\_redis.pref\\_ini](#) sys.argv[2]
- list [mk\\_pgpmac\\_redis.hard\\_ini](#) sys.argv[3]
- dictionary [mk\\_pgpmac\\_redis.configs](#)
- dictionary [mk\\_pgpmac\\_redis.plcc2\\_dict](#)
- dictionary [mk\\_pgpmac\\_redis.motor\\_dict](#)

- dictionary `mk_pgpmac_redis.hard_ini_fields`
- list `mk_pgpmac_redis.motor_field_lists`
- list `mk_pgpmac_redis.bi_list` ["CryoSwitch"]
- dictionary `mk_pgpmac_redis.motor_presets`
- list `mk_pgpmac_redis.zoom_settings`
- tuple `mk_pgpmac_redis.hi iniParser.iniParser( hard_ini)`
- list `mk_pgpmac_redis.v motor_dict[m]`
- string `mk_pgpmac_redis.f "HSETNX"`
- list `mk_pgpmac_redis.xlate hard_ini_fields[k]`
- tuple `mk_pgpmac_redis.pi iniParser.iniParser( pref_ini)`
- int `mk_pgpmac_redis.i 0`
- tuple `mk_pgpmac_redis.ppos pi.get( section, option)`
- string `mk_pgpmac_redis.fnc "HSETNX"`
- tuple `mk_pgpmac_redis.b pi.get( section, "LightIntensity")`
- tuple `mk_pgpmac_redis.p pi.get( section, "MotorPosition")`
- tuple `mk_pgpmac_redis.x pi.get( section, "ScaleX")`
- tuple `mk_pgpmac_redis.y pi.get( section, "ScaleY")`
- tuple `mk_pgpmac_redis.plcc2_file open( "%s-plcc2.pmc" % (head), "w")`
- tuple `mk_pgpmac_redis.motor_num int( motor_dict[m]["motor_num"])`

## 7.11 pgpmac.c File Reference

Main for the pgpmac project.

```
#include "pgpmac.h"
```

### Macros

- `#define PGPMAC_COMMAND_LINE_LENGTH 128`  
*Handle keyboard input.*
- `#define PGPMAC_N_COMMAND_LINES 128`

### Functions

- void `stdinService (struct pollfd *evt)`
- void `pgpmac_printf (char *fmt,...)`  
*Terminal output routine ala printf.*
- void `pgpmac_request_stay_of_execution (int secs)`  
*Postpone the day of reckoning This assumes the quit\_cb routine is called once a second.*
- void `pgpmac_quit_cb (char *event)`  
*quit the program*
- int `main (int argc, char **argv)`  
*Our main routine.*

## Variables

- `WINDOW * term_output`  
*place to print stuff out*
- `WINDOW * term_input`  
*place to put the cursor*
- `WINDOW * term_status`  
*shutter, lamp, air, etc status*
- `WINDOW * term_status2`  
*shutter, lamp, air, etc status*
- `pthread_mutex_t ncurses_mutex`  
*allow more than one thread access to the screen*
- `int doomsday_count = 1`  
*Countdown to quitting time: cleanout routines can request a few more heartbeats.*
- `pthread_mutex_t doomsday_mutex`  
*avoid thread collision while resetting the doomsday clock*
- `int pgpmac_use_pg = 0`  
*non-zero to start up lsgpg thread, 0 to not (reids hash PG in config.HOSTNAME sets this)*
- `int pgpmac_use_autoscint = 0`  
*non-zero to automatically move the alignment stage when the scintillator moves (redis hash AUTOSCINT in config.- HOSTNAME sets this)*
- `static struct pollfd stdinfda`  
*Handle input from the keyboard.*
- `static int running = 1`

### 7.11.1 Detailed Description

Main for the pgpmac project.

#### Date

2012

#### Author

Keith Brister

#### Copyright

All Rights Reserved

Definition in file [pgpmac.c](#).

### 7.11.2 Macro Definition Documentation

#### 7.11.2.1 #define PGPMAC\_COMMAND\_LINE\_LENGTH 128

Handle keyboard input.

Definition at line 258 of file pgpmac.c.

#### 7.11.2.2 #define PGPMAC\_N\_COMMAND\_LINES 128

Definition at line 259 of file pgpmac.c.

### 7.11.3 Function Documentation

#### 7.11.3.1 int main ( int argc, char \*\* argv )

Our main routine.

< argument flags

##### Parameters

in	<i>argc</i>	Number of arguments
in	<i>argv</i>	Vector of argument strings

Definition at line 485 of file pgpmac.c.

```
{
static struct pollfd fda[3];           // input for poll: room for postgres,
                                         pmac, and stdin
static int nfd = 0;                   // number of items in fda
static int pollrtn = 0;
static struct option long_options[] = {
{ "i-vars", 0, NULL, 'i' },
{ "m-vars", 0, NULL, 'm' },
{ NULL,      0, NULL, 0 }
};
int c;
int ivars, mvars;
mvars = 0;
ivars = 0;
int i;                                // standard loop counter
pthread_mutexattr_t mutex_initializer;
static sigset_t our_sigset;

// We use SIGUSR1 but sometimes it's called before a handler is in place so
// we'll just block it
// here and now.
//
sigemptyset( &our_sigset );
sigaddset( &our_sigset, SIGUSR1 );
pthread_sigmask(SIG_BLOCK, &our_sigset, NULL);

while( 1 ) {
    c=getopt_long( argc, argv, "im", long_options, NULL );
    if( c == -1 )
        break;

    switch( c ) {
        case 'i':
            ivars=1;
            break;

        case 'm':
            mvars=1;
            break;
    }
}

stdinfda.fd = 0;
stdinfda.events = POLLIN;

initscr();                                // Start ncurses
raw();                                     // Line buffering disabled, control
                                         chars trapped
keypad( stdscr, TRUE);                    // Why is F1 nifty?
refresh();

// Use recursive mutexes
//
pthread_mutexattr_init( &mutex_initializer );
pthread_mutexattr_settype( &mutex_initializer, PTHREAD_MUTEX_RECURSIVE );

pthread_mutex_init( &ncurses_mutex, &mutex_initializer ); //
    don't lock this mutex yet because we are not multi-threaded until the "_run"
    functions
pthread_mutex_init( &doomsday_mutex, &mutex_initializer );

//
// Since the modules reference objects in other modules it is important
// that everyone is initialized before anyone runs
```

```

//  

// Everyone needs to be able to log messages  

lslogging_init();  

lslogging_run();  

  

// Everyone needs to send and listen for events  

//  

lsevents_init();  

lsevents_run();  

  

//  

// Add a couple of our own  

//  

lsevents_add_listener( "^Quit Program$", pgpmac_quit_cb  

    );  

lsevents_preregister_event( "Quit Program");  

lsevents_preregister_event( "Quitting Program");  

  

//  

// Timers are needed by all too  

//  

lstimer_init();  

lstimer_run();  

  

//  

// Redis is where we get our configuration  

// as well as one of communicating with the outside world  

//  

lsredis_init();  

lsredis_run();  

lsredis_config();  

  

//  

// These need to be all initialized before any are run  

//  

lspmac_init( ivars, mvars);  

  

if( pgpmac_use_pg)  

    lspg_init();  

  

md2cmds_init();  

  

//  

// set up our screen  

//  

pthread_mutex_lock( &ncurses_mutex);  

term_status = newwin( LS_DISPLAY_WINDOW_HEIGHT  

    , LS_DISPLAY_WINDOW_WIDTH, 3*LS_DISPLAY_WINDOW_HEIGHT  

    , 0*LS_DISPLAY_WINDOW_WIDTH);  

box( term_status, 0, 0);  

wnoutrefresh( term_status);  

  

term_status2 = newwin( LS_DISPLAY_WINDOW_HEIGHT  

    , LS_DISPLAY_WINDOW_WIDTH, 3*LS_DISPLAY_WINDOW_HEIGHT  

    , 1*LS_DISPLAY_WINDOW_WIDTH);  

box( term_status2, 0, 0);  

wnoutrefresh( term_status2);  

  

term_output = newwin( 20, 5*LS_DISPLAY_WINDOW_WIDTH  

    , 4*LS_DISPLAY_WINDOW_HEIGHT, 0);  

scrolllok( term_output, 1);  

wnoutrefresh( term_output);  

  

term_input = newwin( 3, 5*LS_DISPLAY_WINDOW_WIDTH  

    , 20+4*LS_DISPLAY_WINDOW_HEIGHT, 0);  

box( term_input, 0, 0);  

mvwprintw( term_input, 1, 1, "PMAC> ");  

nodelay( term_input, TRUE);  

keypad( term_input, TRUE);  

wnoutrefresh( term_input);  

  

doupdate();  

pthread_mutex_unlock( &ncurses_mutex);  

  

//  

// Now run the world  

//  

lspmac_run();  

  

if( pgpmac_use_pg)  

    lspg_run();  

  

md2cmds_run();  

  

while( running) {  

    //  

    // Big loop

```

```

//  

nfd = 0;  

//  

// keyboard  

//  

memcpy( &(fda[nfd++]), &stdinfda, sizeof( struct pollfd));  

  

if( nfd == 0) {  

//  

// No connectons yet. Wait a bit and try again.  

//  

sleep( 10);  

//  

// go try to connect again  

//  

continue;  

}  

  

pollrtn = poll( fda, nfd, 10);  

  

for( i=0; pollrtn>0 && i<nfd; i++) {  

if( fda[i].revents) {  

pollrtn--;  

if( fda[i].fd == 0) {  

stdinService( &fda[i]);  

}  

}  

}  

}  

endwin();  

return 0;  

}

```

### 7.11.3.2 void pgpmac\_printf( char \* fmt, ... )

Terminal output routine ala printf.

#### Parameters

in	fmt	Printf style formating string
----	-----	-------------------------------

Definition at line 443 of file pgpmac.c.

```

{
va_list arg_ptr;  

pthread_mutex_lock( &ncurses_mutex);  

va_start( arg_ptr, fmt);  

vwprintw( term_output, fmt, arg_ptr);  

va_end( arg_ptr);  

wnoutrefresh( term_output);  

doupdate();  

pthread_mutex_unlock( &ncurses_mutex);
}
```

### 7.11.3.3 void pgpmac\_quit\_cb( char \* event )

quit the program

Definition at line 474 of file pgpmac.c.

```

{
pthread_mutex_lock( &doomsday_mutex);
doomsday_count--;
if( doomsday_count <= 0)
running = 0;
pthread_mutex_unlock( &doomsday_mutex);
}
```

#### 7.11.3.4 void pgpmac\_request\_stay\_of\_execution ( int secs )

Postpone the day of reckoning This assumes the quit\_cb routine is called once a second.

Definition at line 464 of file pgpmac.c.

```

{
pthread_mutex_lock( &doomsday_mutex);
if( secs > doomsday_count)
    doomsday_count = secs;
pthread_mutex_unlock( &doomsday_mutex);
}

```

#### 7.11.3.5 void stdInService ( struct pollfd \* evt )

##### Parameters

in	evt	The pollfd object that caused this call
----	-----	---

Definition at line 260 of file pgpmac.c.

```

{
static char cmd_lines[PGPMAC_N_COMMAND_LINES] [
    PGPMAC_COMMAND_LINE_LENGTH];
static int current_line = 0;
static int previous_line = 0;
static char *cmdsp;
static char *prompt = "PMAC>";;
static int cmd_on = 0;
static int cmd_mode = 0;
static char cevt[32];
int ch;
int i;
char tmp;

cmdsp = cmd_lines[current_line];

for( ch=wgetch(term_input); ch != ERR && running; ch=wgetch(
    term_input)) {

    switch( ch) {
    case KEY_F(1):
    case KEY_F(2):
    case KEY_F(3):
        lspmac_abort(); // send
        abort now (as opposed to an event listener) in case a cleanup routine wants to
        move something (we don't want to abort it).
        lsevents_send_event( "Quitting Program");
        // let everyone know the end is nigh
        lstimter_set_timer( "Quit Program", -1, 1, 0); // Doomsday, repeat as needed
        break;

    case 0x0002: // Control-B Report status word for 8 motors
    case 0x0003: // Control-C Report all coordinate system status
        words
    case 0x0006: // Control-F Report following errors for 8 motors
    case 0x0010: // Control-P Report positions for 8 motors
    case 0x0016: // Control-V Report velocity on 8 motors
        sprintf( cevt, "Control-%c", '@' + ch);
        lspmac_SockSendControlCharPrint( cevt, ch)
        ;
        break;

    case 0x0001: // Control-A Abort all programs and moves
    case 0x0004: // Control-D Disable all PLC programs
    case 0x0005: // Control-E Enable disabled motors
    case 0x0007: // Control-G Report global status word
    case 0x000b: // Control-K Kill all motors
    case 0x000f: // Control-O Feed hold on all coordinate systems
    case 0x0011: // Control-Q Quit all executing motion programs
    case 0x0012: // Control-R Run motion programs in all coordinate
        systems
    case 0x0013: // Control-S Step through working motion programs in
        all coordinate systems
    case 0x0018: // Control-X Cancel in-process communications
        sprintf( cevt, "Control-%c", '@' + ch);
        lspmac_SockSendControlCharPrint( cevt, ch)
    }
}

```

```

;
//      lspmac_SockSendDPControlChar( cevt, ch);
break;

case 0x000c:          // Control-L
pthread_mutex_lock( &ncurses_mutex);
redrawwin( term_status);
redrawwin( term_status2);
redrawwin( term_output);
redrawwin( term_input);
for( i=0; i<lspmac_nmotors; i++) {
    if( lspmac_motors[i].win != NULL)
        redrawwin( lspmac_motors[i].win);
}
pthread_mutex_unlock( &ncurses_mutex);
break;

case KEY_UP:
previous_line = (previous_line - 1 + PGPMAC_N_COMMAND_LINES
) % PGPMAC_N_COMMAND_LINES;
if( previous_line == current_line || cmd_lines[previous_line][0] == 0) {
//
// We seem to have gone through all the lines, but NO MORE.
//
previous_line = (previous_line + 1) % PGPMAC_N_COMMAND_LINES
;
}

memset( cmdsp, 0, PGPMAC_COMMAND_LINE_LENGTH);
strcpy( cmdsp, cmd_lines[previous_line]);
cmds_on = strlen(cmdsp);
break;

case KEY_DOWN:
if( previous_line != current_line)
    previous_line = (previous_line + 1) % PGPMAC_N_COMMAND_LINES
;

memset( cmdsp, 0, PGPMAC_COMMAND_LINE_LENGTH);
strcpy( cmdsp, cmd_lines[previous_line]);
cmds_on = strlen(cmdsp);
break;

case KEY_LEFT:
cmds_on = cmds_on == 0 ? 0 : cmds_on - 1;
break;

case KEY_RIGHT:
cmds_on = cmds_on >= strlen(cmdsp) ? strlen(cmdsp) : cmds_on + 1;
break;

case KEY_BACKSPACE:
cmds_on == 0 ? 0 : cmds_on--;
for( i=0; *(cmdsp + cmds_on + i) != 0; i++) {
    *(cmdsp + cmds_on + i) = *(cmdsp + cmds_on + i + 1);
}
break;

case KEY_ENTER:
case 0x000a:
if( cmds_on > 0 && strlen( cmdsp) > 0) {
    switch( cmd_mode) {
    case 0:
        if( strcmp( cmdsp, "$$$") == 0) {
            lsevents_send_event( "Full Card Reset Requested"
);
            lslogging_log_message( "Performing Full Card
Reset, resuming in 10 seconds");
            lstimer_set_timer( "Full Card Reset", 1, 10, 0);
        }
        lspmac_SockSendline( NULL, "%s", cmdsp);
        break;
    case 1:
        md2cmds_push_queue( cmdsp);
        break;
    }
}
current_line = (current_line + 1) % PGPMAC_N_COMMAND_LINES
;
previous_line = current_line;
cmdsp = cmd_lines[current_line];
memset( cmdsp, 0, PGPMAC_COMMAND_LINE_LENGTH);
cmds_on = 0;
break;

default:
if( ch >= 0x20 && ch <= 0x7e) {

```

```

    if( cmdsp_on < PGPMAC_COMMAND_LINE_LENGTH - 1)
    {
        for( i=cmdsp_on; ch != 0 && i < PGPMAC_COMMAND_LINE_LENGTH
; i++) {
            tmp = *(cmdsp + i);
            *(cmdsp + i) = ch;
            ch = tmp;
        }
        cmdsp_on = (cmdsp_on + 1) % PGPMAC_COMMAND_LINE_LENGTH;
    }
}
break;
}

if(strcasecmp( "pmac", cmdsp) == 0) {
    *cmdsp     = 0;
    cmd_mode   = 0;
    cmdsp_on   = 0;
    memset( cmdsp, 0, PGPMAC_COMMAND_LINE_LENGTH);
    prompt = "PMAC>";
}

if(strcasecmp( "md2cmnds", cmdsp) == 0) {
    *cmdsp     = 0;
    cmd_mode   = 1;
    cmdsp_on   = 0;
    memset( cmdsp, 0, PGPMAC_COMMAND_LINE_LENGTH);
    prompt = "md2cmnds>";
}

if( strcasecmp( "quit", cmdsp) == 0) {
    lsmac_abort();                                     // send
    abort now (as opposed to an event listener) in case a cleanup routine wants to
    move something (we don't want to abort it).
    lsevents_send_event( "Quitting Program");
    // let everyone know the end is nigh
    lstimber_set_timer( "Quit Program", -1, 1, 0);      //
    Doomsday, repeat as needed
    *cmdsp     = 0;
    cmdsp_on   = 0;
    memset( cmdsp, 0, PGPMAC_COMMAND_LINE_LENGTH);
}

if( running) {
    pthread_mutex_lock( &ncurses_mutex);
    mvwprintw( term_input, 1, 1, "%s %s", prompt, cmdsp);
    wclrtoeol( term_input);
    wmove( term_input, 1, cmdsp_on + strlen(prompt) + 2);
    box( term_input, 0, 0);
    wnoutrefresh( term_output);
    wnoutrefresh( term_input);
    doupdate();
    pthread_mutex_unlock( &ncurses_mutex);
}
}

```

#### 7.11.4 Variable Documentation

#### 7.11.4.1 int doomsday\_count = 1

Countdown to quitting time: cleanout routines can request a few more heartbeats.

Definition at line 243 of file pgpmac.c.

#### 7.11.4.2 `pthread_mutex_t` doomsday\_mutex

avoid thread collision while resetting the doomsday clock

Definition at line 244 of file pgpmac.c.

#### 7.11.4.3 `pthread_mutex_t` `nurses_mutex`

allow more than one thread access to the screen

Definition at line 242 of file pgpmac.c.

**7.11.4.4 int pgpmac\_use\_autoscint = 0**

non-zero to automatically move the alignment stage when the scintillator moves (redis hash AUTOSCINT in config.-HOSTNAME sets this)

Definition at line 247 of file pgpmac.c.

**7.11.4.5 int pgpmac\_use\_pg = 0**

non-zero to start up lsgp thread, 0 to not (reids hash PG in config.HOSTNAME sets this)

Definition at line 246 of file pgpmac.c.

**7.11.4.6 int running = 1 [static]**

Definition at line 253 of file pgpmac.c.

**7.11.4.7 struct pollfd stdinfda [static]**

Handle input from the keyboard.

Definition at line 252 of file pgpmac.c.

**7.11.4.8 WINDOW\* term\_input**

place to put the cursor

Definition at line 238 of file pgpmac.c.

**7.11.4.9 WINDOW\* term\_output**

place to print stuff out

Definition at line 237 of file pgpmac.c.

**7.11.4.10 WINDOW\* term\_status**

shutter, lamp, air, etc status

Definition at line 239 of file pgpmac.c.

**7.11.4.11 WINDOW\* term\_status2**

shutter, lamp, air, etc status

Definition at line 240 of file pgpmac.c.

## 7.12 pgpmac.h File Reference

Headers for the entire pgpmac project.

```
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netdb.h>
#include <string.h>
#include <netinet/in.h>
#include <errno.h>
#include <poll.h>
#include <libpq-fe.h>
#include <ncurses.h>
#include <math.h>
#include <pthread.h>
#include <signal.h>
#include <sys/signalfd.h>
#include <sys/time.h>
#include <time.h>
#include <getopt.h>
#include <regex.h>
#include <hiredis/hiredis.h>
#include <hiredis/async.h>
#include <search.h>
#include <ctype.h>
```

## Data Structures

- struct [lsredis\\_obj\\_struct](#)  
*Redis Object Basic object whose value is synchronized with our redis db.*
- struct [tagEthernetCmd](#)  
*PMAC ethernet packet definition.*
- struct [lspmac\\_cmd\\_queue\\_struct](#)  
*PMAC command queue item.*
- struct [lspmac\\_motor\\_struct](#)  
*Motor information.*
- struct [lspmac\\_bi\\_struct](#)  
*Storage for binary inputs.*
- struct [lspgQueryQueueStruct](#)  
*Store each query along with its callback function.*
- struct [lspg\\_waitcryo\\_struct](#)
- struct [lspg\\_getcurrentsampleid\\_struct](#)
- struct [lspg\\_demandairrights\\_struct](#)
- struct [lspg\\_getcenter\\_struct](#)  
*Storage for getcenter query Used for the md2 ROTATE command that generates the centering movies.*
- struct [lspg\\_starttransfer\\_struct](#)  
*returns 1 if transfer can continue 0 to abort*
- struct [lspg\\_nextsample\\_struct](#)  
*Returns the next sample number Just a 32 bit int (Ha!, take that, nextshot!)*
- struct [lspg\\_nextshot\\_struct](#)  
*Storage definition for nextshot query.*

## Macros

- `#define _GNU_SOURCE`
- `#define LS_DISPLAY_WINDOW_HEIGHT 8`  
*Number of status box rows.*
- `#define LS_DISPLAY_WINDOW_WIDTH 24`  
*Number of status box columns.*
- `#define LS_PG_QUERY_STRING_LENGTH 1024`  
*Fixed length postgresql query strings. Queries should all be function calls so this is not as weird as one might think.*
- `#define LSEVENTS_EVENT_LENGTH 256`  
*Fixed length for event names: simplifies string handling.*
- `#define LSPMAC_MAGIC_NUMBER 0x9700436`
- `#define MD2CMDS_CMD_LENGTH 32`

## TypeDefs

- `typedef struct lsredis_obj_struct lsredis_obj_t`  
*Redis Object Basic object whose value is synchronized with our redis db.*
- `typedef struct tagEthernetCmd pmac_cmd_t`  
*PMAC ethernet packet definition.*
- `typedef struct lspmac_cmd_queue_struct pmac_cmd_queue_t`  
*PMAC command queue item.*
- `typedef struct lspmac_motor_struct lspmac_motor_t`  
*Motor information.*
- `typedef struct lspmac_bi_struct lspmac_bi_t`  
*Storage for binary inputs.*
- `typedef struct lspgQueryQueueStruct lspg_query_queue_t`  
*Store each query along with its callback function.*
- `typedef struct lspg_waitcryo_struct lspg_waitcryo_t`
- `typedef struct lspg_getcurrentsampleid_struct lspg_getcurrentsampleid_t`
- `typedef struct lspg_demandairrights_struct lspg_demandairrights_t`
- `typedef struct lspg_getcenter_struct lspg_getcenter_t`  
*Storage for getcenter query Used for the md2 ROTATE command that generates the centering movies.*
- `typedef struct lspg_starttransfer_struct lspg_starttransfer_t`  
*returns 1 if transfer can continue 0 to abort*
- `typedef struct lspg_nexsample_struct lspg_nexsample_t`  
*Returns the next sample number Just a 32 bit int (Ha!, take that, nextshot!)*
- `typedef struct lspg_nextshot_struct lspg_nextshot_t`  
*Storage definition for nextshot query.*

## Functions

- `double lsppmac_getPosition (lsppmac_motor_t *)`  
`get the motor position (with locking)`
- `char ** lsppg_array2ptrs (char *)`  
`returns a null terminated list of strings parsed from postgresql array`
- `char ** lsredis_get_string_array (lsredis_obj_t *p)`
- `void lsppmac_SockSendDPLine (char *, char *fmt,...)`  
`prepare (queue up) a line to send the dpram ascii command interface`
- `pmac_cmd_queue_t * lsppmac_SockSendline (char *, char *,...)`  
`Send a one line command.`
- `lsredis_obj_t * lsredis_get_obj (char *,...)`
- `char * lsredis_getstr (lsredis_obj_t *p)`  
`return a copy of the key's string value be sure to free the result`
- `void PmacSockSendline (char *)`
- `unsigned int lsppg_nextsample_all (int *err)`
- `char lsredis_getc (lsredis_obj_t *p)`
- `long int lsredis_getl (lsredis_obj_t *p)`
- `void lsevents_add_listener (char *, void(*cb)(char *))`  
`Add a callback routine to listen for a specific event.`
- `void lsevents_init ()`  
`Initialize this module.`
- `void lsevents_remove_listener (char *, void(*cb)(char *))`  
`Remove a listener previously added with lsevents_add_listener.`
- `void lsevents_run ()`  
`Start up the thread and get out of the way.`
- `void lsevents_send_event (char *,...)`  
`Call the callback routines for the given event.`
- `void lsevents_preregister_event (char *fmt,...)`
- `void lslogging_init ()`  
`Initialize the lslogging objects.`
- `void lslogging_log_message (char *fmt,...)`  
`The routine everyone will be talking about.`
- `void lslogging_run ()`  
`Start up the worker thread.`
- `void lsppg_demandairrights_all ()`  
`do nothing until we get airrights`
- `void lsppg_getcenter_call ()`  
`Request a getcenter query.`
- `void lsppg_getcenter_done ()`  
`Done with getcenter query.`
- `void lsppg_getcenter_wait ()`  
`Wait for a getcenter query to return.`
- `void lsppg_getcurrentsampleid_wait_for_id (unsigned int test)`
- `void lsppg_init ()`  
`Initialize the lsppg module.`
- `void lsppg_nextshot_call ()`  
`Queue up a nextshot query.`
- `void lsppg_nextshot_done ()`  
`Called when the next shot query has been processed.`
- `void lsppg_nextshot_wait ()`

- void **lspg\_query\_push** (void(\*cb)(**lspg\_query\_queue\_t** \*, PGresult \*), char \*fmt,...)
 

*Place a query on the queue.*
- void **lspg\_run** ()
 

*Start 'er runnin'.*
- void **lspg\_seq\_run\_prep\_all** (long long skey, double kappa, double phi, double cx, double cy, double ax, double ay, double az)
 

*Convinience function to call seq run prep.*
- void **lspg\_starttransfer\_call** (unsigned int nextsample, int sample\_detected, double ax, double ay, double az, double horz, double vert, double esttime)
- void **lspg\_starttransfer\_done** ()
- void **lspg\_starttransfer\_wait** ()
 

*no need to get fancy with the wait cryo command It should not return until the robot is almost ready for air rights*
- void **lspg\_waitcryo\_cb** (**lspg\_query\_queue\_t** \*qqp, PGresult \*pgr)
- void **lspg\_zoom\_lut\_call** ()
- int **lspmac\_getBIPosition** (**lspmac\_bi\_t** \*)
 

*get binary input value*
- void **lspmac\_home1\_queue** (**lspmac\_motor\_t** \*mp)
 

*Home the motor.*
- void **lspmac\_home2\_queue** (**lspmac\_motor\_t** \*mp)
 

*Second stage of homing.*
- void **lspmac\_abort** ()
 

*abort motion and try to recover*
- void **lspmac\_init** (int, int)
 

*Initialize this module.*
- int **lspmac\_jogabs\_queue** (**lspmac\_motor\_t** \*, double)
 

*Use jog to move motor to requested position.*
- int **lspmac\_move\_or\_jog\_abs\_queue** (**lspmac\_motor\_t** \*mp, double requested\_position, int use\_jo)
 

*Move method for normal stepper and servo motor objects Returns non-zero on abort, zero if OK.*
- int **lspmac\_move\_or\_jog\_preset\_queue** (**lspmac\_motor\_t** \*, char \*, int)
 

*move using a preset value returns 0 on success, non-zero on error*
- void **lspmac\_move\_or\_jog\_queue** (**lspmac\_motor\_t** \*, double, int)
- int **lspmac\_move\_preset\_queue** (**lspmac\_motor\_t** \*mp, char \*preset\_name)
 

*Move a given motor to one of its preset positions.*
- int **lspmac\_moveabs\_queue** (**lspmac\_motor\_t** \*, double)
 

*Use coordinate system motion program, if available, to move motor to requested position.*
- int **lspmac\_moveabs\_wait** (**lspmac\_motor\_t** \*mp, double timeout)
 

*Wait for motor to finish moving.*
- void **lspmac\_run** ()
 

*Start up the lspmac thread.*
- void **lspmac\_video\_rotate** (double secs)
 

*Special motion program to collect centering video.*
- int **lsredis\_cmpnstr** (**lsredis\_obj\_t** \*p, char \*s, int n)
- int **lsredis\_cmpstr** (**lsredis\_obj\_t** \*p, char \*s)
- int **lsredis\_find\_preset** (char \*base, char \*preset\_name, double \*dval)
 

*Get the value of the given preset and return it in dval Returns 0 on error, non-zero on success;.*
- int **lsredis\_getb** (**lsredis\_obj\_t** \*p)
- double **lsredis\_getd** (**lsredis\_obj\_t** \*p)
- void **lsredis\_init** ()
 

*Initialize this module, that is, set up the connections.*
- int **lsredis\_reexec** (const regex\_t \*preg, **lsredis\_obj\_t** \*p, size\_t nmatch, regmatch\_t \*pmatch, int eflags)

- void `lsredis_run ()`
- void `lsredis_setstr (lsredis_obj_t *p, char *fmt,...)`

*Set the value and update redis.*
- void `lstimter_set_timer (char *, int, unsigned long int, unsigned long int)`

*Create a timer.*
- void `lstimter_unset_timer (char *event)`

*Unsets all timers for the given event.*
- void `lstimter_init ()`

*Initialize the timer list and pthread stuff.*
- void `lstimter_run ()`

*Start up our thread.*
- void `lsupdate_init ()`
- void `lsupdate_run ()`
- void `md2cmds_init ()`

*Initialize the md2cmds module.*
- void `md2cmds_run ()`

*Start up the thread.*
- void `pgpmac_printf (char *fmt,...)`

*Terminal output routine ala printf.*
- void `lptest_main ()`
- int `lspmac_est_move_time (double *est_time, int *mmask, lspmac_motor_t *mp_1, int jog_1, char *preset_1, double end_point_1,...)`

*Move the motors and estimate the time it'll take to finish the job.*
- int `lspmac_est_move_time_wait (double move_time, int cmask, lspmac_motor_t *mp_1,...)`

*wait for motion to stop returns non-zero if the wait timed out*
- void `lsredis_set_preset (char *base, char *preset_name, double dval)`

*set the given preset to the given value create a new preset if we can't find it*
- `lsredis_obj_t * lsredis_get_obj (char *key)`

*Maybe add a new object Used internally for this module Must be called with lsredis\_mutex locked.*
- `lspmac_motor_t * lspmac_find_motor_by_name (char *name)`
- int `lsredis_find_preset_index_by_position (lspmac_motor_t *mp)`

*For the given motor object return the index of the current preset or -1 if we are not at a preset position.*
- void `lspmac_SockSendDPCControlChar (char *event, char c)`

*use dpram ascii interface to send a control character*
- int `lspmac_set_motion_flags (int *mmaskp, lspmac_motor_t *mp_1,...)`

*Set the coordinate system motion flags (m5075) for the null terminated list of motors that we are planning on running a motion program with.*
- void `lsredis_load_presets (char *motor_name)`

*update the presets hash table for the named motor*
- void `pgpmac_request_stay_of_execution (int secs)`

*Postpone the day of reckoning This assumes the quit\_cb routine is called once a second.*
- void `md2cmds_push_queue (char *action)`
- `pmac_cmd_queue_t * lspmac_SockSendControlCharPrint (char *event, char c)`

*Send a control character.*
- void `lsredis_config ()`

## Variables

- `lspg_waitcryo_t` `lspg_waitcryo`  
*signal the robot*
- `lspg_getcurrentsampleid_t` `lspg_getcurrentsampleid`  
*our currentsample id*
- `lspg_demandairrights_t` `lspg_demandairrights`  
*our demandairrights object*
- `lspg_getcenter_t` `lspg_getcenter`  
*the getcenter object*
- `lspg_starttransfer_t` `lspg_starttransfer`  
*start a sample transfer*
- `lspg_nextsample_t` `lspg_nextsample`  
*the very next sample*
- `int pgpmac_use_pg`  
*non-zero to start up lspg thread, 0 to not (reids hash PG in config.HOSTNAME sets this)*
- `int pgpmac_use_autoscint`  
*non-zero to automatically move the alignment stage when the scintillator moves (redis hash AUTOSCINT in config.-HOSTNAME sets this)*
- `lspg_nextshot_t` `lspg_nextshot`  
*the nextshot object*
- `lspmac_motor_t` `lspmac_motors []`  
*All our motors.*
- `lspmac_motor_t *` `omega`  
*MD2 omega axis (the air bearing)*
- `lspmac_motor_t *` `alignx`  
*Alignment stage X.*
- `lspmac_motor_t *` `aligny`  
*Alignment stage Y.*
- `lspmac_motor_t *` `alignz`  
*Alignment stage Z.*
- `lspmac_motor_t *` `anal`  
*Polaroid analyzer motor.*
- `lspmac_motor_t *` `zoom`  
*Optical zoom.*
- `lspmac_motor_t *` `apery`  
*Aperture Y.*
- `lspmac_motor_t *` `aperz`  
*Aperture Z.*
- `lspmac_motor_t *` `capy`  
*Capillary Y.*
- `lspmac_motor_t *` `capz`  
*Capillary Z.*
- `lspmac_motor_t *` `scint`  
*Scintillator Z.*
- `lspmac_motor_t *` `cenx`  
*Centering Table X.*
- `lspmac_motor_t *` `ceny`  
*Centering Table Y.*
- `lspmac_motor_t *` `kappa`  
*Kappa.*

- `lspmac_motor_t * phi`  
*Phi (not data collection axis)*
- `lspmac_motor_t * fshut`  
*Fast shutter.*
- `lspmac_motor_t * flight`  
*Front Light DAC.*
- `lspmac_motor_t * blight`  
*Back Light DAC.*
- `lspmac_motor_t * fscint`  
*Scintillator Piezo DAC.*
- `lspmac_motor_t * smart_mag_oo`  
*Smart Magnet on/off.*
- `lspmac_motor_t * blight_ud`  
*Back light Up/Down actuator.*
- `lspmac_motor_t * cryo`  
*Move the cryostream towards or away from the crystal.*
- `lspmac_motor_t * dryer`  
*blow air on the scintilator to dry it off*
- `lspmac_motor_t * fluo`  
*Move the fluorescence detector in/out.*
- `lspmac_motor_t * flight_oo`  
*Turn front light on/off.*
- `lspmac_motor_t * blight_f`  
*Back light scale factor.*
- `lspmac_motor_t * flight_f`  
*Front light scale factor.*
- `int lspmac_nmotors`  
*The number of motors we manage.*
- `lspmac_bi_t * lp_air`  
*Low pressure air OK.*
- `lspmac_bi_t * hp_air`  
*High pressure air OK.*
- `lspmac_bi_t * cryo_switch`  
*that little toggle switch for the cryo*
- `lspmac_bi_t * blight_down`  
*Backlight is down.*
- `lspmac_bi_t * blight_up`  
*Backlight is up.*
- `lspmac_bi_t * cryo_back`  
*cryo is in the back position*
- `lspmac_bi_t * fluor_back`  
*fluor is in the back position*
- `lspmac_bi_t * sample_detected`  
*smart magnet detected sample*
- `lspmac_bi_t * etel_ready`  
*ETEL is ready.*
- `lspmac_bi_t * etel_on`  
*ETEL is on.*
- `lspmac_bi_t * etel_init_ok`  
*ETEL initialized OK.*
- `lspmac_bi_t * minikappa_ok`

- Minikappa is OK (whatever that means)*
- `lspmac.bi_t * smart_mag_on`
  - smart magnet is on*
- `lspmac.bi_t * arm_parked`
  - (whose arm? parked where?)*
- `lspmac.bi_t * shutter_open`
  - shutter is open (note in pmc says this is a slow input)*
- `lspmac.bi_t * smart_mag_off`
  - smart magnet is off*
- `lspmac.bi_t * smart_mag_err`
  - smart magnet error (coil broken perhaps)*
- struct timespec `omega_zero_time`
  - Time we believe that omega crossed zero.*
- WINDOW \* `term_output`
  - place to print stuff out*
- WINDOW \* `term_input`
  - place to put the cursor*
- WINDOW \* `term_status`
  - shutter, lamp, air, etc status*
- WINDOW \* `term_status2`
  - shutter, lamp, air, etc status*
- pthread\_mutex\_t `ncurses_mutex`
  - allow more than one thread access to the screen*
- pthread\_cond\_t `md2cmds_cond`
  - condition to signal when it's time to run an md2 command*
- pthread\_mutex\_t `md2cmds_mutex`
  - mutex for the condition*
- pthread\_cond\_t `md2cmds_pg_cond`
- pthread\_mutex\_t `md2cmds_pg_mutex`
- pthread\_mutex\_t `pmac_queue_mutex`
  - manage access to the pmac command queue*
- pthread\_cond\_t `pmac_queue_cond`
  - wait for a command to be sent to PMAC before continuing*
- pthread\_mutex\_t `lspmac_shutter_mutex`
  - Coordinates threads reading shutter status.*
- pthread\_cond\_t `lspmac_shutter_cond`
  - Allows waiting for the shutter status to change.*
- int `lspmac_shutter_state`
  - State of the shutter, used to detect changes.*
- int `lspmac_shutter_has_opened`
  - Indicates that the shutter had opened, perhaps briefly even if the state did not change.*
- pthread\_mutex\_t `lspmac_moving_mutex`
  - Coordinate moving motors between threads.*
- pthread\_cond\_t `lspmac_moving_cond`
  - Wait for motor(s) to finish moving condition.*
- int `lspmac_moving_flags`
  - Flag used to implement motor moving condition.*
- pthread\_mutex\_t `md2_status_mutex`
  - Synchronize reading/writing status buffer.*
- char `md2cmds_cmd []`
  - our command;*

- `lsredis_obj_t * md2cmds_md_status_code`
- `pthread_mutex_t lsredis_mutex`
- `pthread_cond_t lsredis_cond`
- `int lsredis_running`

### 7.12.1 Detailed Description

Headers for the entire pgpmac project.

#### Date

2012

#### Author

Keith Brister

#### Copyright

All Rights Reserved

Definition in file [pgpmac.h](#).

### 7.12.2 Macro Definition Documentation

#### 7.12.2.1 #define \_GNU\_SOURCE

Definition at line 7 of file pgpmac.h.

#### 7.12.2.2 #define LS\_DISPLAY\_WINDOW\_HEIGHT 8

Number of status box rows.

Definition at line 57 of file pgpmac.h.

#### 7.12.2.3 #define LS\_DISPLAY\_WINDOW\_WIDTH 24

Number of status box columns.

Definition at line 61 of file pgpmac.h.

#### 7.12.2.4 #define LS\_PG\_QUERY\_STRING\_LENGTH 1024

Fixed length postgresql query strings. Queries should all be function calls so this is not as weird as one might think.

Definition at line 64 of file pgpmac.h.

#### 7.12.2.5 #define LSEVENTS\_EVENT\_LENGTH 256

Fixed length for event names: simplifies string handling.

Definition at line 67 of file pgpmac.h.

---

7.12.2.6 `#define LSPMAC_MAGIC_NUMBER 0x9700436`

Definition at line 95 of file pgpmac.h.

7.12.2.7 `#define MD2CMDS_CMD_LENGTH 32`

Definition at line 491 of file pgpmac.h.

### 7.12.3 Typedef Documentation

7.12.3.1 `typedef struct lsgp_demandairrights_struct lsgp_demandairrights_t`

7.12.3.2 `typedef struct lsgp_getcenter_struct lsgp_getcenter_t`

Storage for getcenter query Used for the md2 ROTATE command that generates the centering movies.

7.12.3.3 `typedef struct lsgp_getcurrentsampleid_struct lsgp_getcurrentsampleid_t`

7.12.3.4 `typedef struct lsgp_nexsample_struct lsgp_nexsample_t`

Returns the next sample number Just a 32 bit int (Hal!, take that, nextshot!)

7.12.3.5 `typedef struct lsgp_nextshot_struct lsgp_nextshot_t`

Storage definition for nextshot query.

The next shot query returns all the information needed to collect the next data frame. Since SQL allows for null fields independently from blank strings a separate integer is used as a flag for this case. This adds to the program complexity but allows for some important cases. Suck it up.definition of the next image to be taken (and the one after that, too!)

7.12.3.6 `typedef struct lsgpQueryQueueStruct lsgp_query_queue_t`

Store each query along with its callback function.

All calls are asynchronous

7.12.3.7 `typedef struct lsgp_starttransfer_struct lsgp_starttransfer_t`

returns 1 if transfer can continue 0 to abort

7.12.3.8 `typedef struct lsgp_waitcryo_struct lsgp_waitcryo_t`

7.12.3.9 `typedef struct lspmac_bi_struct lspmac_bi_t`

Storage for binary inputs.

7.12.3.10 `typedef struct lspmac_motor_struct lspmac_motor_t`

Motor information.

A catchall for motors and motor like objects. Not all members are used by all objects.

7.12.3.11 **typedef struct lsredis\_obj\_struct lsredis\_obj\_t**

Redis Object Basic object whose value is synchronized with our redis db.

7.12.3.12 **typedef struct lsmpmac\_cmd\_queue\_struct pmac\_cmd\_queue\_t**

PMAC command queue item.

Command queue items are fixed length to simplify memory management.

7.12.3.13 **typedef struct tagEthernetCmd pmac\_cmd\_t**

PMAC ethernet packet definition.

Taken directly from the Delta Tau documentation.

## 7.12.4 Function Documentation

7.12.4.1 **lsredis\_obj\_t\* \_lsredis\_get\_obj( char \* key )**

Maybe add a new object Used internally for this module Must be called with lsredis\_mutex locked.

Definition at line 510 of file lsredis.c.

```

{
    lsredis_obj_t *p;
    regmatch_t pmatch[2];
    int err;
    ENTRY htab_input, *htab_output;

    // Dispense with obviously bad keys straight away
    // unless p->valid == 0 in which case we call HGET first
    //
    // TODO: review logic: is there ever a time when valid is zero for a
    // preexisting p and HGET has not been called?
    //      If not then we should just return p without checking for validity.
    //
    if( key == NULL || *key == 0 || strchr( key, ' ' ) != NULL) {
        lslogging_log_message( "_lsredis_get_obj: bad key '%s'
            ", key == NULL ? "<NULL>" : key);
        return NULL;
    }

    // If the key is already there then just return it
    //

    htab_input.key = key;
    htab_input.data = NULL;
    errno = 0;
    err = hsearch_r( htab_input, FIND, &htab_output, &lsredis_htab);

    if( err == 0)
        p = NULL;
    else
        p = htab_output->data;

    if( p != NULL) {
        return p;
    } else {
        // make a new one.
        p = calloc( 1, sizeof( lsredis_obj_t));
        if( p == NULL) {
            lslogging_log_message( "_lsredis_get_obj: Out of
                memory");
            exit( -1);
        }

        err = regexec( &lsredis_key_select_regex, key, 2,
            pmatch, 0);
        if( err == 0 && pmatch[1].rm_so != -1) {
            p->events_name = strndup( key+pmatch[1].rm_so, pmatch[1].rm_eo
                - pmatch[1].rm_so);
        } else {

```

```

    p->events_name = strdup( key );
}
if( p->events_name == NULL) {
    lslogging_log_message( "_lsredis_get_obj: Out of
        memory (events_name)");
    exit( -1);
}

pthread_mutex_init( &p->mutex, &mutex_initializer);
pthread_cond_init( &p->cond, NULL);
p->value = NULL;
p->valid = 0;
lsevents_send_event( "%s Invalid", p->events_name
    );
p->wait_for_me = 0;
p->key = strdup( key );
p->hits = 0;

htab_input.key = p->key;
htab_input.data = p;

errno = 0;
err = hsearch_r( htab_input, ENTER, &htab_output, &lsredis_htab
    );
if( err == 0) {
    lslogging_log_message( "_lsredis_get_obj: hsearch
        error on enter. errno=%d", errno);
}

/*
// Shouldn't need the linked list unless we need to rebuild the hash table
when, for example, we run out of room.
// TODO: resize hash table when needed.
//
p->next = lsredis_objs;
lsredis_objs = p;
}
*/
// We arrive here with the valid flag lowered. Go ahead and request the
latest value.
//
redisAsyncCommand( roac, lsredis_hgetCB, p, "HGET %s VALUE"
    , key);

return p;
}

```

#### 7.12.4.2 void lsevents\_add\_listener( char \* raw\_regex, void()(char \*) cb )

Add a callback routine to listen for a specific event.

##### Parameters

<i>raw_regex</i>	String value of regular expression to listen to
<i>cb</i>	the routine to call

Definition at line 99 of file lsevents.c.

```

{
lsevents_listener_t    *new;
lsevents_event_names_t *enp;
lsevents_callbacks_t   *cbp;
int err;
char *errbuf;
int nerrbuf;

new = calloc( 1, sizeof( lsevents_listener_t));
if( new == NULL) {
    lslogging_log_message( "lsevents_add_listener: out of
        memory");
    exit( -1);
}

err = regcomp( &new->re, raw_regex, REG_EXTENDED | REG_NOSUB);
if( err != 0) {
    nerrbuf = regerror( err, &new->re, NULL, 0);
    errbuf = calloc( nerrbuf, sizeof( char));

```

```

    if( errbuf == NULL) {
        lslogging_log_message( "lsevents_add_listener: out
            of memory (re)");
        exit( -1);
    }
    regerror( err, &new->re, errbuf, nerrbuf);
    //    lslogging_log_message( "lsevents_add_listener: %s", errbuf);
    free( errbuf);
    free( new);
    return;
}

new->raw_regexp = strdup( raw_regexp);
new->cb      = cb;

pthread_mutex_lock( &lsevents_listener_mutex);
new->next = lsevents_listeners_p;
lsevents_listeners_p = new;

for( enp = lsevents_event_names; enp != NULL; enp = enp->
    next) {
    if( regexec( &new->re, enp->event, 0, NULL, 0) == 0) {
        cbp      = calloc( 1, sizeof( lsevents_callbacks_t))
        ;
        cbp->cb      = cb;
        cbp->next = enp->cbl;
        enp->cbl = cbp;
    }
}

pthread_mutex_unlock( &lsevents_listener_mutex);

//  lslogging_log_message( "lsevents_add_listener: added listener for event
'%' , raw_regexp);
}

```

#### 7.12.4.3 void lsevents\_init( )

Initialize this module.

Definition at line 373 of file lsevents.c.

```

{
pthread_mutexattr_t mutex_initializer;

// Use recursive mutexes
//
pthread_mutexattr_init( &mutex_initializer);
pthread_mutexattr_settype( &mutex_initializer, PTHREAD_MUTEX_RECURSIVE);

pthread_mutex_init( &lsevents_queue_mutex,   //
    mutex_initializer);
pthread_cond_init( &lsevents_queue_cond,     NULL);
pthread_mutex_init( &lsevents_listener_mutex, &
    mutex_initializer);

hcreate_r( 2*lsevents_max_events, &lsevents_event_name_ht
    );
}

```

#### 7.12.4.4 void lsevents\_preregister\_event( char \* fmt, ... )

Definition at line 314 of file lsevents.c.

```

{
char s[128];
va_list arg_ptr;

va_start( arg_ptr, fmt);
vsnprintf( s, sizeof( s) - 1, fmt, arg_ptr);
s[sizeof(s)-1] = 0;
va_end( arg_ptr);

lsevents_register_event( s);
}

```

#### 7.12.4.5 void lsevents\_remove\_listener( char \* event, void(\*)(char \*) cb )

Remove a listener previously added with lsevents\_add\_listener.

##### Parameters

<i>event</i>	The name of the event (possibly a regular expression string)
<i>cb</i>	The callback routine to remove

Definition at line 157 of file lsevents.c.

```
{
lsevents_listener_t *last, *current;
lsevents_event_names_t *enp;
lsevents_callbacks_t *cbp, *last_cbp;

// Find the listener to remove
// and unlink it from the list
//
pthread_mutex_lock( &lsevents_listener_mutex );
last = NULL;
for( current = lsevents_listeners_p; current != NULL;
    current = current->next ) {
    if( strcmp( last->raw_regexp, event ) == 0 && last->cb == cb) {
        if( last == NULL) {
            lsevents_listeners_p = current->next;
        } else {
            last->next = current->next;
        }
        break;
    }
    last = current;
}

if( current == NULL) {
    lslogging_log_message( "lsevents_remove_listener:
        Could not find this listener for event '%s'", event);
    pthread_mutex_unlock( &lsevents_listener_mutex );
    return;
}

// Remove callback from lists of event names
//
for( enp = lsevents_event_names; enp != NULL; enp = enp->
    next) {
    if( regexec( &current->re, enp->event, 0, NULL, 0) == 0) {
        last_cbp = NULL;
        for( cbp = enp->cbl; cbp != NULL; cbp = cbp->next) {
            if( cbp->cb == cb) {
                if( last_cbp == NULL)
                    enp->cbl = NULL;
                else
                    last_cbp->next = cbp->next;
                free( cbp);
                break;
            }
        }
    }
}

pthread_mutex_unlock( &lsevents_listener_mutex );

// Now remove it
//
if( current->raw_regexp != NULL)
    free( current->raw_regexp);
free( current);
}
```

#### 7.12.4.6 void lsevents\_run( )

Start up the thread and get out of the way.

Definition at line 390 of file lsevents.c.

```
{
pthread_create( &lsevents_thread, NULL, lsevents_worker
, NULL);
}
```

#### 7.12.4.7 void lsevents\_send\_event( char \* fmt, ... )

Call the callback routines for the given event.

##### Parameters

<i>fmt</i>	a printf style formating string
...	list of arguments specified by the format string

Definition at line 73 of file lsevents.c.

```
{
char event[LSEVENTS_EVENT_LENGTH];
va_list arg_ptr;

va_start( arg_ptr, fmt);
vsnprintf( event, sizeof(event)-1, fmt, arg_ptr);
event[sizeof(event)-1]=0;
va_end( arg_ptr);

pthread_mutex_lock( &lsevents_queue_mutex);

// maybe wait for room on the queue
while( (lsevents_queue_on + 1) % LSEVENTS_QUEUE_LENGTH
== lsevents_queue_off % LSEVENTS_QUEUE_LENGTH
)
pthread_cond_wait( &lsevents_queue_cond, &
lsevents_queue_mutex);

lsevents_queue[(lsevents_queue_on++) %
LSEVENTS_QUEUE_LENGTH].evp = strdup(event);

pthread_cond_signal( &lsevents_queue_cond);
pthread_mutex_unlock( &lsevents_queue_mutex);
}
```

#### 7.12.4.8 void lslogging\_init( )

Initialize the lslogging objects.

Definition at line 37 of file lslogging.c.

```
{
pthread_mutex_init( &lslogging_mutex, NULL);
pthread_cond_init( &lslogging_cond, NULL);

lslogging_file = fopen( LSLOGGING_FILE_NAME,
"w");
}
```

#### 7.12.4.9 void lslogging\_log\_message( char \* fmt, ... )

The routine everyone will be talking about.

##### Parameters

<i>fmt</i>	A printf style formating string.
...	The arguments specified by fmt

Definition at line 48 of file lslogging.c.

```

    {
char msg[LSLOGGING_MSG_LENGTH];
struct timespec theTime;
va_list arg_ptr;
unsigned int on;

clock_gettime( CLOCK_REALTIME, &theTime);

va_start( arg_ptr, fmt);
vsnprintf( msg, sizeof(msg)-1, fmt, arg_ptr);
va_end( arg_ptr);
msg[sizeof(msg)-1]=0;

pthread_mutex_lock( &lslogging_mutex);

on = (lslogging_on++) % LSLOGGING_QUEUE_LENGTH
;
strncpy( lslogging_queue[on].lmsg, msg, LSLOGGING_MSG_LENGTH
- 1);
lslogging_queue[on].lmsg[LSLOGGING_MSG_LENGTH
-1] = 0;

memcpy( &(lslogging_queue[on].ltime), &theTime, sizeof(theTime
));
}

pthread_cond_signal( &lslogging_cond);
pthread_mutex_unlock( &lslogging_mutex);
}

```

#### 7.12.4.10 void lslogging\_run( )

Start up the worker thread.

Definition at line 121 of file lslogging.c.

```

    {
pthread_create( &lslogging_thread, NULL, &lslogging_worker
, NULL);
lslogging_log_message( "Start up");
lsevents_add_listener( ".+", lslogging_event_cb
);
}

```

#### 7.12.4.11 char\*\* lspg\_array2ptrs( char \* )

returns a null terminated list of strings parsed from postgresql array

Definition at line 160 of file lspg.c.

```

    {
char **rtn, *sp, *acums;
int i, n, inquote, havebackslash, rtini;;
int mxsz;

inquote = 0;
havebackslash = 0;

// Despense with the null input condition before we complicate the code below
if( a == NULL || a[0] != '{' || a[strlen(a)-1] != '}')
return NULL;

// Count the maximum number of strings
// Actual number will be less if there are quoted commas
//
n = 1;
for( i=0; a[i]; i++) {
    if( a[i] == ',')
        n++;
}
//
// The maximum size of any string is the length of a (+1)
//
```

```

mksz = strlen(a) + 1;

// This is the accumulation string to make up the array elements
acums = (char *)calloc( mksz, sizeof( char));
if( acums == NULL) {
    lslogging_log_message( "lspg_array2ptrs: out of memory
        (acums)");
    exit( 1);
}

//
// allocate storage for the pointer array and the null terminator
//
rtn = (char **)calloc( n+1, sizeof( char *));
if( rtn == NULL) {
    lslogging_log_message( "lspg_array2ptrs: out of memory
        (rtn)");
    exit( 1);
}
rtni = 0;

// Go through and create the individual strings
sp = acums;
*sp = 0;

inquote = 0;
havebackslash = 0;
for( i=1; a[i] != 0; i++) {
    switch( a[i]) {
    case '\"':
        if( havebackslash) {
            // a quoted quote.  Cool
            //
            *(sp++) = a[i];
            *sp = 0;
            havebackslash = 0;
        } else {
            // Toggle the flag
            inquote = 1 - inquote;
        }
        break;

    case '\\':
        if( havebackslash) {
            *(sp++) = a[i];
            *sp = 0;
            havebackslash = 0;
        } else {
            havebackslash = 1;
        }
        break;

    case ',':
        if( inquote || havebackslash) {
            *(sp++) = a[i];
            *sp = 0;
            havebackslash = 0;
        } else {
            rtn[rtni++] = strdup( acums);
            sp = acums;
        }
        break;

    case ')':
        if( inquote || havebackslash) {
            *(sp++) = a[i];
            *sp = 0;
            havebackslash = 0;
        } else {
            rtn[rtni++] = strdup( acums);
            rtn[rtni] = NULL;
            free( acums);
            return( rtn);
        }
        break;

    default:
        *(sp++) = a[i];
        *sp = 0;
        havebackslash = 0;
    }
}

//
// Getting here means the final ')' was missing
// Probably we should throw an error or log it or something.
// Through out the last entry since this there is not resonable expectation
// that

```

```
// we should be parsing it anyway.
//
rtn[rtni] = NULL;
free( acums);
return( rtn);
}
```

#### 7.12.4.12 void lsgc\_demandairrights\_all( )

do nothing until we get airrights

Definition at line 647 of file lsgc.c.

```
{
lsgc_demandairrights_call();
lsgc_demandairrights_wait();
// there is no "done" version
}
```

#### 7.12.4.13 void lsgc\_getcenter\_call( )

Request a getcenter query.

Definition at line 1268 of file lsgc.c.

```
{
pthread_mutex_lock( &lsgc_getcenter.mutex);
lsgc_getcenter.new_value_ready = 0;
pthread_mutex_unlock( &lsgc_getcenter.mutex);

lsgc_query_push( lsgc_getcenter_cb, "SELECT *
    FROM px.getcenter2()");
}
```

#### 7.12.4.14 void lsgc\_getcenter\_done( )

Done with getcenter query.

Definition at line 1286 of file lsgc.c.

```
{
pthread_mutex_unlock( &(lsgc_getcenter.mutex));
}
```

#### 7.12.4.15 void lsgc\_getcenter\_wait( )

Wait for a getcenter query to return.

Definition at line 1278 of file lsgc.c.

```
{
pthread_mutex_lock( &(lsgc_getcenter.mutex));
while( lsgc_getcenter.new_value_ready == 0)
    pthread_cond_wait( &(lsgc_getcenter.cond), &(
        lsgc_getcenter.mutex));
}
```

## 7.12.4.16 void lsgp\_getcurrentsampleid\_wait\_for\_id( unsigned int test )

Definition at line 483 of file lsgp.c.

```

    {
pthread_mutex_lock( &lsgp_getcurrentsampleid.mutex
    );
while( lsgp_getcurrentsampleid.getcurrentsampleid
!= test)
pthread_cond_wait( &lsgp_getcurrentsampleid.cond
, &lsgp_getcurrentsampleid.mutex);

pthread_mutex_unlock( &lsgp_getcurrentsampleid.mutex
);
}
}
```

## 7.12.4.17 void lsgp\_init( )

Initialize the lsgp module.

Definition at line 1979 of file lsgp.c.

```

{
pthread_mutex_init( &lsgp_queue_mutex, NULL);
pthread_cond_init( &lsgp_queue_cond, NULL);

lsgp_demandairrights_init();
lsgp_getcenter_init();
lsgp_getcurrentsampleid_init();
lsgp_lock_detector_init();
lsgp_lock_diffractometer_init();
lsgp_nextsample_init();
lsgp_nextshot_init();
lsgp_seq_run_prep_init();
lsgp_starttransfer_init();
lsgp_wait_for_detector_init();
lsgp_waitcryo_init();
}
}
```

## 7.12.4.18 unsigned int lsgp\_nextsample\_all( int \*err )

Definition at line 558 of file lsgp.c.

```

{
unsigned int rtn;

lsgp_nextsample_call();
lsgp_nextsample_wait();

if( lsgp_nextsample.no_rows_returned) {
    rtn = 0;
    *err = 1;
} else {
    if( lsgp_nextsample.nextsample_isnull) {
        rtn = 0;
        *err = 1;
    } else {
        rtn = lsgp_nextsample.nextsample;
        *err = 0;
    }
}
lsgp_nextsample_done();

return rtn;
}
```

## 7.12.4.19 void lsgp\_nextshot\_call( )

Queue up a nextshot query.

Definition at line 915 of file lsgp.c.

```

    {
    pthread_mutex_lock( &(lspg_nextshot.mutex));
    lspg_nextshot.new_value_ready = 0;
    pthread_mutex_unlock( &(lspg_nextshot.mutex));

    lspg_query_push( lspg_nextshot_cb, "SELECT *
        FROM px.nextshot2()");
}

```

#### 7.12.4.20 void lspg\_nextshot\_done( )

Called when the next shot query has been processed.

Definition at line 933 of file lspg.c.

```

    {
    pthread_mutex_unlock( &(lspg_nextshot.mutex));
}

```

#### 7.12.4.21 void lspg\_nextshot\_wait( )

Wait for the next shot query to get processed.

Definition at line 925 of file lspg.c.

```

    {
    pthread_mutex_lock( &(lspg_nextshot.mutex));
    while( lspg_nextshot.new_value_ready == 0)
        pthread_cond_wait( &(lspg_nextshot.cond), &(lspg_nextshot
            .mutex));
}

```

#### 7.12.4.22 void lspg\_query\_push( void(\*)(lspg\_query\_queue\_t \*, PGresult \*) cb, char \* fmt, ... )

Place a query on the queue.

##### Parameters

in	<i>cb</i>	Our callback function that deals with the response
in	<i>fmt</i>	Printf style function to generate the query

Definition at line 127 of file lspg.c.

```

    {
    int idx;
    va_list arg_ptr;

    pthread_mutex_lock( &lspg_query_mutex);

    // Pause the thread while we service the queue
    //
    while( (lspg_query_queue_on + 1) %
        LS_PG_QUERY_QUEUE_LENGTH == lspg_query_queue_off %
        LS_PG_QUERY_QUEUE_LENGTH) {
        pthread_cond_wait( &lspg_query_cond, &lspg_query_mutex
            );
    }

    idx = lspg_query_queue_on % LS_PG_QUERY_QUEUE_LENGTH
        ;

    va_start( arg_ptr, fmt);
    vsnprintf( lspg_query_queue[idx].qs,
        LS_PG_QUERY_STRING_LENGTH-1, fmt, arg_ptr);
    va_end( arg_ptr);

    lspg_query_queue[idx].qs[LS_PG_QUERY_STRING_LENGTH

```

```

    - 1] = 0;
lspg_query_queue[idx].onResponse = cb;
lspg_query_queue_on++;

pthread_kill( lspg_thread, SIGUSR1);
pthread_mutex_unlock( &lspg_queue_mutex);
};
}

```

#### 7.12.4.23 void lspg\_run( )

Start 'er runnin'.

Definition at line 1998 of file Lspg.c.

```

{
pthread_create( &lspg_thread, NULL, lspg_worker, NULL);
lsevents_add_listener( "^(appy|appz|capy|capz|scint) In
Position$",
lspg_check_preset_in_position_cb);
lsevents_add_listener( "^(appy|appz|capy|capz|scint)
Moving$",
lspg_unset_current_preset_moving_cb
);
lsevents_add_listener( "^Preset Changed (.+)",
lspg_preset_changed_cb);
lsevents_add_listener( "^Sample(Detected|Absent)$",
lspg_sample_detector_cb);
lsevents_add_listener( "^Timer Update KV$",
lspg_update_kvs_cb);
lsevents_add_listener( "^cam.zoom In Position$",
lspg_set_scale_cb);
lstopper_set_timer( "Timer Update KV", -1, 0, 500000000)
;

// // Make sure we own the airrights
// lspg_demandairrights_all();
}

```

#### 7.12.4.24 void lspg\_seq\_run\_prep\_all( long long skey, double kappa, double phi, double cx, double cy, double ax, double ay, double az )

Convinence function to call seq run prep.

##### Parameters

in	<i>skey</i>	px.shots key for this image
in	<i>kappa</i>	current kappa position
in	<i>phi</i>	current phi position
in	<i>cx</i>	current center table x
in	<i>cy</i>	current center table y
in	<i>ax</i>	current alignment table x
in	<i>ay</i>	current alignment table y
in	<i>az</i>	current alignment table z

Definition at line 1186 of file Lspg.c.

```

{
lspg_seq_run_prep_call( skey, kappa, phi, cx,
cy, ax, ay, az);
lspg_seq_run_prep_wait();
lspg_seq_run_prep_done();
}

```

**7.12.4.25 void lsgp\_starttransfer\_call ( unsigned int nextsample, int sample\_detected, double ax, double ay, double az, double horz, double vert, double esttime )**

Definition at line 389 of file lsgp.c.

```
{
pthread_mutex_lock( &(lsgp_starttransfer.mutex));
lsgp_starttransfer.new_value_ready = 0;
pthread_mutex_unlock( &(lsgp_starttransfer.mutex));

lsgp_query_push( lsgp_starttransfer_cb,
    "SELECT px.starttransfer( %d, %s, %.3f, %.3f, %.3f, %.3f, %.3f, %.3f )",
    nextsample, sample_detected ? "True" : "False
    ", ax, ay, az, horz, vert, esttime);
}
```

**7.12.4.26 void lsgp\_starttransfer\_done ( )**

Definition at line 404 of file lsgp.c.

```
{
pthread_mutex_unlock( &(lsgp_starttransfer.mutex));
}
```

**7.12.4.27 void lsgp\_starttransfer\_wait ( )**

Definition at line 398 of file lsgp.c.

```
{
pthread_mutex_lock( &(lsgp_starttransfer.mutex));
while( lsgp_starttransfer.new_value_ready ==
    0)
pthread_cond_wait( &(lsgp_starttransfer.cond), &
    lsgp_starttransfer.mutex);
}
```

**7.12.4.28 void lsgp\_waitcryo\_all ( )**

no need to get fancy with the wait cryo command It should not return until the robot is almost ready for air rights

Definition at line 597 of file lsgp.c.

```
{
pthread_mutex_lock( &lsgp_waitcryo.mutex);
lsgp_waitcryo.new_value_ready = 0;

lsgp_query_push( lsgp_waitcryo_cb, "SELECT
px.waitcryo()");

while( lsgp_waitcryo.new_value_ready == 0)
pthread_cond_wait( &lsgp_waitcryo.cond, &lsgp_waitcryo
.mutex);

pthread_mutex_unlock( &lsgp_waitcryo.mutex);
}
```

**7.12.4.29 void lsgp\_waitcryo\_cb ( lsgp\_query\_queue\_t \* qqp, PGresult \* pgr )**

Definition at line 587 of file lsgp.c.

```
{
pthread_mutex_lock( &lsgp_waitcryo.mutex);
lsgp_waitcryo.new_value_ready = 1;
pthread_cond_signal( &lsgp_waitcryo.cond);
pthread_mutex_unlock( &lsgp_waitcryo.mutex);
}
```

7.12.4.30 void lsgp\_zoom\_lut\_call( )

7.12.4.31 void lspmact\_abort( )

abort motion and try to recover

Definition at line 2104 of file lspmact.c.

```

{
// Stop everything! (consider ^O instead of ^A)
// lspmact_SockSendDPControlChar( "Abort Request", 0
x01);

// and reset motion flag
// lspmact_SockSendDPLine( "Reset", "%s", "M5075=0");

}

```

7.12.4.32 int lspmact\_est\_move\_time( double \* est\_time, int \* mmaskp, lspmact\_motor\_t \* mp\_1, int jog\_1, char \* preset\_1, double end\_point\_1, ... )

Move the motors and estimate the time it'll take to finish the job.

Returns the estimate time and the coordinate system mask to wait for

#### Parameters

<i>est_time</i>	Returns number of seconds we estimate the move(s) will take
<i>mmaskp</i>	Mask of coordinate systems we are trying to move, excluding jogs. Used to wait for motions to complete
<i>mp_1</i>	Pointer to first motor
<i>jog_1</i>	1 to force a jog, 0 to try a motion program DO NOT MIX JOGS AND MOTION PROGRAMS IN THE SAME COORDINATE SYSTEM!
<i>preset_1</i>	Name of preset we'd like to move to or NULL if <i>end_point_1</i> should be used instead
<i>end_point_1</i>	End point for the first motor. Ignored if <i>preset_1</i> is non null and identifies a valid preset for this motor
...	Perhaps more quads of motors, jog flags, preset names, and end points. End is a NULL motor pointer MUST END ARG LIST WITH NULL

< units to counts

< The total distance we need to go

< Our maximum velocity

< Our maximum acceleration

< Total time for this motor

< coordinate system motion flags

Definition at line 2829 of file lspmact.c.

```

static char axes[] = "XYZUVWABC";
int qs[9];
lspmact_combined_move_t motions[32];
char s[256];
int foundone;
int moving_flags;
struct timespec timeout;
int j;
va_list arg_ptr;
lspmact_motor_t *mp;
{
```

```

double ep, maybe_ep;
char *ps;
double
min_pos,
max_pos,
neutral_pos,
u2c,
D,
V,
A,
Tt;
int err;
int jog;
int i;
uint32_t m5075;

// reset our coordinate flags and command strings
//
for( i=0; i<32; i++) {
    motions[i].moveme = 0;
}
m5075 = 0;
if( mmaskp != NULL)
    *mmaskp = 0;

//
// Initialize first iteration
//
*est_time = 0.0;
mp = mp_1;
ps = preset_1;
ep = end_point_1;
jog = jog_1;

va_start( arg_ptr, end_point_1);
while( 1) {
/*
 *      :           | Constant          |
 *      :           |<-- Velocity   --->|
 *      :           |           Time (Ct)   |
 * V :-----+
 * e :           / \-----+
 * l :           / \-----+
 * o :           / \-----+
 * c :           / \-----+
 * i :           / \-----+
 * t :           / \-----+
 * y :-----+-----+
 *          |           | Time
 *          |           |-----+
 *          -->|           |<-- Acceleration Time (At)
 *          |           |-----+
 *          |<----- Total Time (Tt) ----->|
 *
 *      Assumption 1: We can replace S curve acceleration with linear
 *      acceleration
 *      for the purposes of distance and time calculations for the timeout
 *      period that we are attempting to calculate here.
 *
 *      Ct = Constant Velocity Time. The time spent at constant velocity.
 *
 *      At = Acceleration Time. Time spent accelerating at either end of
 *      the ramp, that is,
 *      1/2 the total time spent accelerating and decelerating.
 *
 *      D = the total distance we need to travel
 *
 *      V = constant velocity. Here we use the motor's maximum velocity.
 *
 *      A = the motor acceleration, Here it's the maximum acceleration.
 *
 *      V = A * At
 *
 *      or At = V/A
 *
 *      The Total Time (Tt) is
 *
 *      Tt = Ct + 2 * At
 *
 *
 *      If we had infinite acceleration the total time would be D/V. To
 *      account for finite acceleration we just need to
 *      adjust this for the average velocity while accelerating (0.5 V).
 *      This neatly adds a single V/A term:
 */
}

```

```

*      (1)      Tt = D/V + V/A
*
*      When the distance is short, we need a different calculation:
*
*      D = 0.5 * A * T1^2 + 0.5 * A * T2^2 (T1 = acceleration time and
*      T2 = deceleration time)
*
*      or, since total time Tt = T1 + T2 and T1 = T2,
*
*      D = A * (0.5*Tt)^2
*
*      or
*
*      (2)      Tt = 2 * sqrt( D/A)
*
*
*      When we accelerate to the maximum speed the time it takes is V/A so
*      the distance we travel (Da) is
*
*      Da = 0.5 * A * (V/A)^2
*
*      or
*
*      Da = 0.5 * V^2 / A
*
*      So when D > 2 * Da, or
*
*      D > V^2 / A
*
*      we need to use equation (1) otherwise we need to use equation (2)
*
*/
}

if( mp->magic != LSMPMAC_MAGIC_NUMBER) {
    lslogging_log_message( "lspmac_est_move_time:
        WARNING: bad motor structure. Check that your motor list is NULL terminated.");
    break;
}

lslogging_log_message( "lspmac_est_move_time: find
    motor %s, jog %d, preset %s, endpoint %f",
    mp->name, jog, ps == NULL ? "NULL" : ps, ep);

Tt = 0.0;
if( mp != NULL && mp->max_speed != NULL && mp->max_accel
!= NULL && mp->u2c != NULL) {

    //
    // get the real endpoint if a preset was mentioned
    //
    if( ps != NULL && *ps != 0) {
        err = lsredis_find_preset( mp->name, ps, &
        maybe_ep);
        if( err != 0)
            ep = maybe_ep;
    }

    u2c = lsredis_getd( mp->u2c);

    //
    // For look up tables user units are (or should be) counts and u2c should
    be 1
    //
    if( mp->nlut > 0 && mp->lut != NULL) {
        u2c = 1.0;
        D = lspmac_lut( mp->nlut, mp->lut, ep) - lspmac_lut
        ( mp->nlut, mp->lut, lspmacGetPosition( mp));
    } else {
        D = ep - lspmacGetPosition( mp);
        // User units
    }

    V = lsredis_getd( mp->max_speed) / u2c * 1000.;
    // User units per second
    A = lsredis_getd( mp->max_accel) / u2c * 1000. *
    1000;           // User units per second per second

    neutral_pos = lsredis_getd( mp->neutral_pos);
    min_pos     = lsredis_getd( mp->min_pos) - neutral_pos
    ;
    max_pos     = lsredis_getd( mp->max_pos) - neutral_pos
    ;

    if( ep < min_pos || ep > max_pos) {
        lslogging_log_message( "lspmac_est_move_time:

```

```

Motor %s Requested position %f out of range: min=%f, max=%f", mp->name,
min_pos, max_pos);
    lsevents_send_event( "%s Move Aborted", mp->name
);
    return 1;
}

mp->requested_position = ep;
mp->requested_pos_cnts = u2c * (mp->requested_position
+ neutral_pos);

//  

// Don't bother with motors without velocity or acceleration defined  

//  

if( V > 0.0 && A > 0.0 ) {
    if( fabs(D) > V*V/A ) {
        //  

        // Normal ramp up, constant velocity, and ramp down  

        //  

        Tt = fabs(D)/V + V/A;
    } else {
        //  

        // Never reach constantant velocity, just ramp up a bit and back down  

        //  

        Tt = 2.0 * sqrt( fabs(D)/A );
    }
}

lslogging_log_message( "lspmac_est_move_time:  

Motor: %s D: %f VV/A: %f Tt: %f", mp->name, D, V*V/A, Tt);
} else {
    //  

    // TODO: insert move time based for DAC or BO motor like objects;  

    // For now assume 100 msec;  

    //  

    Tt = 0.1;
}

// Perhaps flag a coordinate system
//  

// We can move a motor that's not in a coordinate system but we cannot
move a motor that is but does not
// have an axis defined if we are also moving one that does. It's a
limitation, I guess.
//  

if( jog != 1 &&
    mp->coord_num != NULL && lsredis_get1( mp->
coord_num ) > 0 && lsredis_get1( mp->coord_num ) <=
16 &&
    mp->motor_num != NULL && lsredis_get1( mp->
motor_num ) > 0 && mp->axis != NULL && lsredis_getc( mp
->axis ) != 0 ) {
    int axis;
    int motor_num;

    motor_num = lsredis_get1( mp->motor_num );

    axis = lsredis_getc( mp->axis );
    for( j=0; j<sizeof(axes); j++ ) {
        if( axis == axes[j] )
            break;
    }

    if( j < sizeof( axes) ) {
        //  

        // Store the motion request for a normal PMAC motor
        //  

        int cn;
        int in_position_band;

        cn = lsredis_get1( mp->coord_num );
        in_position_band = lsredis_get1( mp->in_position_band
);

        motions[motor_num - 1].coord_num = cn;
        motions[motor_num - 1].axis = j;
        motions[motor_num - 1].Delta = D * u2c;
        //  

        // Don't ask to run a motion program if we are already where we want
to be
        //  

        // Deadband is 10 counts except for zoom which is 100.
        // We use Ixx28 In-Position Band which has units of 1/16 count
        //  

        //  

        if( abs(motions[motor_num - 1].Delta)*16 >= in_position_band ) {
            m5075 |= (1 << (cn - 1));
    }
}
}

```

```

        motions[motor_num - 1].moveme      = 1;
    }
    lslogging_log_message( "lspmac_est_move_time:
    moveme=%d motor '%s' motions index=%d coord_num=%d axis=%d Delta=%d m5075=%u",
    motions[motor_num-1].moveme, mp->name,
    motor_num -1, motions[motor_num-1].coord_num, motions[motor_num-1].axis
    , motions[motor_num-1].Delta,
    m5075);
}
} else {
//
// Here we are dealing with a DAC or BO motor or just want to jog.
//
if( mp->jogAbs( mp, ep)) {
    lslogging_log_message( "lspmac_est_move_time:
motor %s failed to queue move of distance %f from %f", mp->name, D,
lspmac_getPosition(mp));
    lsevents_send_event( "Move Aborted");
    return 1;
}
//
// Update the estimated time
//
*est_time = *est_time < Tt ? Tt : *est_time;

lslogging_log_message( "lspmac_est_move_time:
est_time=%f", *est_time);
}

mp = va_arg( arg_ptr, lspmac_motor_t *);
if( mp == NULL)
break;

jog = va_arg( arg_ptr, int);
ps  = va_arg( arg_ptr, char *);
ep  = va_arg( arg_ptr, double);

}
va_end( arg_ptr);

// Set the motion program flags
//
if( m5075 != 0) {
if( mmaskp != NULL)
*mmaskp |= m5075; // Tell the caller about our new mask

pthread_mutex_lock( &lspmac_moving_mutex);
moving_flags = lspmac_moving_flags;
pthread_mutex_unlock( &lspmac_moving_mutex);

if( (moving_flags & m5075) != m5075) {
    lspmac_SockSendDLine( NULL, "M5075=(M5075 | %d)",
m5075);

    pthread_mutex_lock( &lspmac_moving_mutex);
    clock_gettime( CLOCK_REALTIME, &timeout);
    //
    timeout.tv_sec += 2;      // 2 seconds should be more than enough time to
    set the flags
    err = 0;
    while( err == 0 && ((lspmac_moving_flags & m5075) !=
m5075))
        err = pthread_cond_timedwait( &lspmac_moving_cond, &
lspmac_moving_mutex, &timeout);
    moving_flags = lspmac_moving_flags;
    pthread_mutex_unlock( &lspmac_moving_mutex);

    if( ((moving_flags & m5075) != m5075) && err == ETIMEDOUT) {
        lslogging_log_message( "lspmac_est_move_time:
Timed out waiting for moving flags. lspmac_moving_flags = 0x%0x, looking for 0x%0x
test exp: 0x%0x test: %d",
        moving_flags, m5075, (moving_flags & m5075), (
        moving_flags & m5075) != m5075);
        lsevents_send_event( "Combined Move Aborted");
        return 1;
    }
}

for( i=1; i<=16; i++) {
}

```

```

// Loop over coordinate systems
//
foundone = 0;

for( j=0; j<9; j++)
    qs[j] = 0;

for( j=0; j<31; j++) {
    //
    // Loop over motors
    //
    if( motions[j].moveme && motions[j].coord_num == i) {
        if( abs(motions[j].Delta) > 0 ) {
            qs[(int)(motions[j].axis)] = motions[j].Delta;
            foundone=1;
        }
    }
}

if( foundone) {
    sprintf( s, "%d Q40=%d Q41=%d Q42=%d Q43=%d Q44=%d Q45=%d Q46=%d Q47=%d
    Q48=%d Q49=%lf Q100=%d B180R",
            i, qs[0], qs[1], qs[2], qs[3], qs[4], qs[5], qs[6], qs[7], qs[8]
        , *est_time * 1000., 1 << (i-1));
    lsmac_SockSendDPLine( NULL, s );
}

return 0;
}

```

```
7.12.4.33 int lspmac_est_move_time_wait ( double move_time, int cmask, lspmac_motor_t * mp_1, ... )
```

wait for motion to stop returns non-zero if the wait timed out

## Parameters

<i>move_time</i>	The time out in seconds
<i>cmask</i>	A coordinate system mask to wait for
<i>mp_1</i>	NULL terminated list of individual motors to wait for

Both values are returned from `lspmac_est_move_time`

Definition at line 3184 of file Ispmac.c.

```

int err;
double isecs, fsecs;
struct timespec timeout;
va_list arg_ptr;
lspmac_motor_t *mp;

clock_gettime( CLOCK_REALTIME, &timeout);
fsecs = modf( move_time, &isecs);
timeout.tv_sec  += (long)floor(isecs);
timeout.tv_nsec += (long)floor(fsecs * 1.e9);
timeout.tv_sec  += timeout.tv_nsec / 1000000000;
timeout.tv_nsec %= 1000000000;

err = 0;
pthread_mutex_lock( &lspmac_moving_mutex);
while( err == 0 && (lspmac_moving_flags & cmask) != 0)
  err = pthread_cond_timedwait( &lspmac_moving_cond, &
    lspmac_moving_mutex, &timeout);
pthread_mutex_unlock( &lspmac_moving_mutex);

if( err != 0) {
  if( err == ETIMEDOUT) {
    lslogging_log_message( "
      ltest_lspmac_est_move_time_wait: timed out waiting %f seconds, cmask = 0x%0x", move_time, cmask);
  }
  lspmac_abort();
  return 1;
}

```

```

va_start( arg_ptr, mp_1);
for( mp = mp_1; mp != NULL; mp = va_arg( arg_ptr, lspmac_motor_t
*) ) {
    if( mp->magic != LSPMAC_MAGIC_NUMBER) {
        lslogging_log_message( "lspmac_est_move_time_wait:
        WARNING: motor list must be NULL terminated. Check your call to
        lspmac_est_move_time_wait.");
    }

    if( lspmac_moveabs_wait( mp, move_time)) {
        lslogging_log_message( "lspmac_est_move_time_wait:
        timed out waiting %f seconds for motor %s", move_time, mp->name);
        return 1;
    }
}
va_end( arg_ptr);

return 0;
}

```

#### 7.12.4.34 lspmac\_motor\_t\* lspmac\_find\_motor\_by\_name ( char \* name )

Definition at line 4311 of file lspmac.c.

```

{
lspmac_motor_t *rtn;
ENTRY entry_in, *entry_outp;
int err;

entry_in.key = name;
entry_in.data = NULL;
err = hsearch_r( entry_in, FIND, &entry_outp, &motors_ht);
if( err == 0) {
    lslogging_log_message( "lspmac_find_motor_by_name:
    hsearch_r failed for motor '%s': %s", name, strerror( errno));
    return NULL;
}
rtn = entry_outp->data;

return rtn;
}

```

#### 7.12.4.35 int lspmac\_getBIPosition ( lspmac\_bi\_t \* )

get binary input value

Definition at line 1649 of file lspmac.c.

```

{
int rtn;
pthread_mutex_lock( &bip->mutex);
rtn = bip->position;
pthread_mutex_unlock( &bip->mutex);
return rtn;
}

```

#### 7.12.4.36 double lspmacGetPosition ( lspmac\_motor\_t \* mp )

get the motor position (with locking)

##### Parameters

<i>mp</i>	the motor object
-----------	------------------

Definition at line 1402 of file lspmac.c.

```

double rtn;
{

```

```

pthread_mutex_lock( &(mp->mutex));
rtn = mp->position;
pthread_mutex_unlock( &(mp->mutex));
return rtn;
}

```

#### 7.12.4.37 void lspmact\_home1\_queue ( lspmact\_motor\_t \* mp )

Home the motor.

##### Parameters

in	<i>mp</i>	motor we are concerned about
----	-----------	------------------------------

Definition at line 1272 of file lspmact.c.

```

{
int i;
int motor_num;
int coord_num;
char **home;

pthread_mutex_lock( &(mp->mutex));

motor_num = lsredis_get1( mp->motor_num);
coord_num = lsredis_get1( mp->coord_num);
home      = lsredis_get_string_array( mp->home);

// Each of the motors should have this defined
// but let's not seg fault if home is missing
//
if( home == NULL || *home == NULL) {
//
// Note we are already initialized
// so if we are here there is something wrong.
//
lslogging_log_message( "lspmact_home1_queue: null or
empty home strings for motor %s", mp->name);
pthread_mutex_unlock( &(mp->mutex));
return;
}

// We've already been called. Don't home again until
// we're finish with the last time.
//
if( mp->homming) {
pthread_mutex_unlock( &(mp->mutex));
return;
}

//
// Don't go on if any other motors in this coordinate system are homing.
// It's possible to write the homing program to home all the motors in the
// coordinate
// system. TODO (hint hint)
//
if( coord_num > 0) {
for( i=0; i<lspmact_nmotors; i++) {
if( &(lspmact_motors[i]) == mp)
continue;
if( lsredis_get1(lspmact_motors[i].coord_num) ==
coord_num) {
int nogo;
nogo = 0;
pthread_mutex_lock( &(lspmact_motors[i].mutex));
//
// Don't go on if
//
// we are homing      or      ( not in position
while      in open loop)
//
if( lspmact_motors[i].homming || (((lspmact_motors
[i].status2 & 0x01)==0) && ((lspmact_motors[i].status1 & 0x040000)
!= 0)))
nogo = 1;
pthread_mutex_unlock( &(lspmact_motors[i].mutex));
if( nogo) {
pthread_mutex_unlock( &(mp->mutex));
return;
}
}
}
}

```

```

        }
    }
}

mp->homing = 1;
mp->not_done = 1;      // set up waiting for cond
mp->motion_seen = 0;
// This opens the control loop.
// The status routine should notice this and the fact that
// the homing flag is set and call on the home2 routine
//
// Only send the open loop command if we are not in
// open loop mode already. This test might prevent a race condition
// where we've already moved the home2 routine (and queue the homing program
// motion)
// before the open loop command is dequeued and acted on.
//
if( ~(mp->status1) & 0x040000) {
    lspmac_SockSendDLine( mp->name, "#%d$*",
    motor_num);
}

pthread_mutex_unlock( &(mp->mutex));
lsevents_send_event( "%s Homing", mp->name);
}

```

#### 7.12.4.38 void lspmac\_home2\_queue ( lspmac\_motor\_t \* mp )

Second stage of homing.

##### Parameters

in	<i>mp</i>	motor we are concerned about
----	-----------	------------------------------

Definition at line 1360 of file lspmac.c.

```

{
char **spp;
char **home;

// At this point we are in open loop.
// Run the motor specific commands
//

pthread_mutex_lock( &(mp->mutex));

home = lsredis_get_string_array( mp->home);

//
// We don't have any motors that have a null home text array so
// there is currently no need to worry about this case other than
// not to seg fault
//
// Also, Only go on if the first homing phase has been started
//
if( home == NULL || mp->homing != 1) {
    pthread_mutex_unlock( &(mp->mutex));
    return;
}

for( spp = home; *spp != NULL; spp++) {

    lslogging_log_message( "home2 is queuing '%s'\n", *spp
    );

    lspmac_SockSendDLine( mp->name, *spp);
}

mp->homing = 2;
pthread_mutex_unlock( &(mp->mutex));
}

```

### 7.12.4.39 void lspmac\_init( int, int )

Initialize this module.

Definition at line 3799 of file lspmac.c.

```

{
static int first_time = 1;
int i;
int err;
ENTRY entry_in, *entry_outp;
md2_status_t *p;
pthread_mutexattr_t mutex_initializer;

if( first_time) {
    // Set our global harvest flags
    getivars = ivarsflag;
    getmvars = mvarsflag;

    // Use recursive mutexs
    //
    pthread_mutexattr_init( &mutex_initializer);
    pthread_mutexattr_settype( &mutex_initializer, PTHREAD_MUTEX_RECURSIVE);

    // All important status mutex
    pthread_mutex_init( &md2_status_mutex, &mutex_initializer);

    //
    // Get the MD2 initialization strings
    //
    // lspmac_md2_init = lsredis_get_obj( "md2_pmac.init"); // hard coded
    now.

    //
    // Initialize the motor objects
    //

    p = &md2_status;

    omega = lspmac_motor_init( &(lspmac_motors
        [ 0]), 0, 0, &p->omega_act_pos,      &p->omega_status_1
        , &p->omega_status_2,      "Omega #1 &1 X", "omega",
        lspmac_moveabs_queue, lspmac_jogabs_queue
        );
    alignx = lspmac_motor_init( &(lspmac_motors
        [ 1]), 0, 1, &p->alignx_act_pos,      &p->alignx_status_1
        , &p->alignx_status_2,      "Align X #2 &3 X", "align.x",
        lspmac_moveabs_queue, lspmac_jogabs_queue
        );
    aligny = lspmac_motor_init( &(lspmac_motors
        [ 2]), 0, 2, &p->aligny_act_pos,      &p->aligny_status_1
        , &p->aligny_status_2,      "Align Y #3 &3 Y", "align.y",
        lspmac_moveabs_queue, lspmac_jogabs_queue
        );
    alignz = lspmac_motor_init( &(lspmac_motors
        [ 3]), 0, 3, &p->alignz_act_pos,      &p->alignz_status_1
        , &p->alignz_status_2,      "Align Z #4 &3 Z", "align.z",
        lspmac_moveabs_queue, lspmac_jogabs_queue
        );
    anal = lspmac_motor_init( &(lspmac_motors
        [ 4]), 0, 4, &p->analyzer_act_pos,      &p->analyzer_status_1
        , &p->analyzer_status_2,      "Anal #5", "lightPolar",
        lspmac_moveabs_queue, lspmac_jogabs_queue
        );
    zoom = lspmac_motor_init( &(lspmac_motors
        [ 5]), 1, 0, &p->zoom_act_pos,      &p->zoom_status_1,
        &p->zoom_status_2,      "Zoom #6 &4 Z", "cam.zoom",
        lspmac_movezoom_queue, lspmac_movezoom_queue
        );
    apery = lspmac_motor_init( &(lspmac_motors
        [ 6]), 1, 1, &p->aperturey_act_pos, &p->aperturey_status_1
        , &p->aperturey_status_2, "Aper Y #7 &5 Y", "appy",
        lspmac_moveabs_queue, lspmac_jogabs_queue
        );
    aperz = lspmac_motor_init( &(lspmac_motors
        [ 7]), 1, 2, &p->aperturez_act_pos, &p->aperturez_status_1
        , &p->aperturez_status_2, "Aper Z #8 &5 Z", "appz",
        lspmac_moveabs_queue, lspmac_jogabs_queue
        );
    capy = lspmac_motor_init( &(lspmac_motors
        [ 8]), 1, 3, &p->capy_act_pos,      &p->capy_status_1,
        &p->capy_status_2,      "Cap Y #9 &5 U", "capy",
        lspmac_moveabs_queue, lspmac_jogabs_queue
        );
    capz = lspmac_motor_init( &(lspmac_motors

```

```

[ 9]), 1, 4, &p->capz_act_pos,      &p->capz_status_1,
    &p->capz_status_2,      "Cap Z #10 &5 V", "capz",
    lspmac_moveabs_queue, lspmac_jogabs_queue
);
scint = lspmac_motor_init( &(lspmac_motors
[10]), 2, 0, &p->scint_act_pos,      &p->scint_status_1
,      &p->scint_status_2,      "Scin Z #11 &5 W", "scint",
    lspmac_moveabs_queue, lspmac_jogabs_queue
);
cenx = lspmac_motor_init( &(lspmac_motors
[11]), 2, 1, &p->centerx_act_pos,      &p->centerx_status_1
,      &p->centerx_status_2,      "Cen X #17 &2 X", "centering.x",
    lspmac_moveabs_queue, lspmac_jogabs_queue
);
ceny = lspmac_motor_init( &(lspmac_motors
[12]), 2, 2, &p->centery_act_pos,      &p->centery_status_1
,      &p->centery_status_2,      "Cen Y #18 &2 Y", "centering.y",
    lspmac_moveabs_queue, lspmac_jogabs_queue
);
kappa = lspmac_motor_init( &(lspmac_motors
[13]), 2, 3, &p->kappa_act_pos,      &p->kappa_status_1
,      &p->kappa_status_2,      "Kappa #19 &7 X", "kappa",
    lspmac_moveabs_queue, lspmac_jogabs_queue
);
phi = lspmac_motor_init( &(lspmac_motors
[14]), 2, 4, &p->phi_act_pos,      &p->phi_status_1
,      &p->phi_status_2,      "Phi #20 &7 Y", "phi",
    lspmac_moveabs_queue, lspmac_jogabs_queue
);

fshut = lspmac_fshut_init( &(lspmac_motors
[15]));
flight = lspmac_dac_init( &(lspmac_motors
[16]), &p->front_dac,      "M1200", "frontLight.intensity",
    lspmac_movedac_queue);
blight = lspmac_dac_init( &(lspmac_motors
[17]), &p->back_dac,      "M1201", "backLight.intensity",
    lspmac_movedac_queue);
fscint = lspmac_dac_init( &(lspmac_motors
[18]), &p->scint_piezo,      "M1203", "scint.focus",
    lspmac_movedac_queue);

smart_mag_oo = lspmac_bo_init( &(lspmac_motors
[19]), "smartMagnet", "M1100=%d", &(md2_status.accllc_5), 0x01)
;
blight_ud = lspmac_bo_init( &(lspmac_motors
[20]), "backLight",      "M1101=%d", &(md2_status.accllc_5), 0x02)
;
cryo = lspmac_bo_init( &(lspmac_motors
[21]), "cryo",      "M1102=%d", &(md2_status.accllc_5), 0x04)
;
dryer = lspmac_bo_init( &(lspmac_motors
[22]), "dryer",      "M1103=%d", &(md2_status.accllc_5), 0x08)
;
fluo = lspmac_bo_init( &(lspmac_motors
[23]), "fluo",      "M1104=%d", &(md2_status.accllc_5), 0x10)
;
flight_oo = lspmac_soft_motor_init( &
    lspmac_motors[24]), "frontLight",
    lspmac_moveabs_frontlight_oo_queue);
blight_f = lspmac_soft_motor_init( &
    lspmac_motors[25]), "backLight.factor",
    lspmac_moveabs_blight_factor_queue);
flight_f = lspmac_soft_motor_init( &
    lspmac_motors[26]), "frontLight.factor",
    lspmac_moveabs_flight_factor_queue);

lp_air = lspmac_bi_init( &(lspmac_bis
[ 0]), &(md2_status.accllc_1), 0x01, "Low Pressure Air OK",
    "Low Pressure Air Failed");
hp_air = lspmac_bi_init( &(lspmac_bis
[ 1]), &(md2_status.accllc_1), 0x02, "High Pressure Air OK",
    "High Pressure Air Failed");
cryo_switch = lspmac_bi_init( &(lspmac_bis
[ 2]), &(md2_status.accllc_1), 0x04, "CryoSwitchChanged",
    "CryoSwitchChanged");
blight_down = lspmac_bi_init( &(lspmac_bis
[ 3]), &(md2_status.accllc_1), 0x08, "Backlight Down",
    "Backlight Not Down");
blight_up = lspmac_bi_init( &(lspmac_bis
[ 4]), &(md2_status.accllc_1), 0x10, "Backlight Up",
    "Backlight Not Up");
cryo_back = lspmac_bi_init( &(lspmac_bis
[ 5]), &(md2_status.accllc_1), 0x40, "Cryo Back",
    "Cryo Not Back");
fluor_back = lspmac_bi_init( &(lspmac_bis
[ 6]), &(md2_status.accllc_2), 0x01, "Fluor. Det. Parked",
    "Fluor. Det. Failed");

```

```

    "Fluor. Det. Not Parked");
sample_detected = lspmac_bi_init( &(lspmac_bis
[ 7]), &(md2_status.accl1c_2), 0x02, "SamplePresent",
"SampleAbsent");
etel_ready = lspmac_bi_init( &(lspmac_bis
[ 8]), &(md2_status.accl1c_2), 0x20, "ETEL Ready",
"ETEL Not Ready");
etel_on = lspmac_bi_init( &(lspmac_bis
[ 9]), &(md2_status.accl1c_2), 0x40, "ETEL On",
"ETEL Off");
etel_init_ok = lspmac_bi_init( &(lspmac_bis
[10]), &(md2_status.accl1c_2), 0x80, "ETEL Init OK",
"ETEL Init Not OK");
minikappa_ok = lspmac_bi_init( &(lspmac_bis
[11]), &(md2_status.accl1c_3), 0x01, "Minikappa OK",
"Minikappa Not OK");
smart_mag_on = lspmac_bi_init( &(lspmac_bis
[12]), &(md2_status.accl1c_3), 0x04, "Smart Magnet On",
"Smart Magnet Not On");
arm_parked = lspmac_bi_init( &(lspmac_bis
[13]), &(md2_status.accl1c_3), 0x08, "Arm Parked",
"Arm Not Parked");
smart_mag_err = lspmac_bi_init( &(lspmac_bis
[14]), &(md2_status.accl1c_3), 0x10, "Smart Magnet Error",
"Smart Magnet OK");
shutter_open = lspmac_bi_init( &(lspmac_bis
[15]), &(md2_status.accl1c_3), 0x100, "Shutter Open",
"Shutter Not Open");
smart_mag_off = lspmac_bi_init( &(lspmac_bis
[16]), &(md2_status.accl1c_5), 0x01, "Smart Magnet Off",
"Smart Magnet Not Off");

// Set up hash table
//
err = hcreate_r( LSPMAC_MAX_MOTORS * 2, &motors_ht
);
if( err == 0) {
    lslogging_log_message( "lspmac_init: hcreate_r
    failed: '%s'", strerror( errno));
    exit( -1);
}
for( i=0; i<lspmac_nmotors; i++) {
    entry_in.key = lspmac_motors[i].name;
    entry_in.data = &(lspmac_motors[i]);
    err = hsearch_r( entry_in, ENTER, &entry_outp, &motors_ht);
    if( err == 0) {
        lslogging_log_message( "lspmac_init: hsearch_r
        failed for motor %s: '%s'", lspmac_motors[i].name, strerror( errno));
        exit( -1);
    }
}

// Initialize several commands that get called, perhaps, alot
//
rr_cmd.RequestType = VR_UPLOAD;
rr_cmd.Request = VR_PMAC_READREADY;
rr_cmd.wValue = 0;
rr_cmd.wIndex = 0;
rr_cmd.wLength = htons(2);
memset( rr_cmd.bData, 0, sizeof(rr_cmd.bData));

gb_cmd.RequestType = VR_UPLOAD;
gb_cmd.Request = VR_PMAC_GETBUFFER;
gb_cmd.wValue = 0;
gb_cmd.wIndex = 0;
gb_cmd.wLength = htons(1400);
memset( gb_cmd.bData, 0, sizeof(gb_cmd.bData));

cr_cmd.RequestType = VR_UPLOAD;
cr_cmd.Request = VR_CTRL_RESPONSE;
cr_cmd.wValue = 0;
cr_cmd.wIndex = 0;
cr_cmd.wLength = htons(1400);
memset( cr_cmd.bData, 0, sizeof(cr_cmd.bData));

//
// Initialize some mutexes and conditions
//

pthread_mutex_init( &pmac_queue_mutex, &mutex_initializer);
pthread_cond_init( &pmac_queue_cond, NULL);

```

```

lspmac_shutter_state = 0;
// assume the shutter is now closed: not a big deal if we are wrong
pthread_mutex_init( &lspmac_shutter_mutex, &
    mutex_initializer);
pthread_cond_init( &lspmac_shutter_cond, NULL);
pmacfd.fd = -1;

pthread_mutex_init( &lspmac_moving_mutex, &
    mutex_initializer);
pthread_cond_init( &lspmac_moving_cond, NULL);

pthread_mutex_init( &lspmac_ascii_mutex, &
    mutex_initializer);

pthread_mutex_init( &lspmac_ascii_buffers_mutex,
    &mutex_initializer);

lsevents_preregister_event( "omega crossed zero")
;
lsevents_preregister_event( "Move Aborted");
lsevents_preregister_event( "Combined Move
    Aborted");
lsevents_preregister_event( "Abort Request queued
    ");
lsevents_preregister_event( "Abort Request
    accepted");
lsevents_preregister_event( "Reset queued");
lsevents_preregister_event( "Reset command
    accepted");

for( i=1; i<=16; i++) {
    lsevents_preregister_event( "Coordsys %d
        Stopped", i);
}
first_time = 0;
}
//
// clear the ascii communications buffers
//
{
    uint32_t cc;
    cc = 0;
    lspmac_send_command( VR_UPLOAD, VR_PMAC_SETMEM
        , 0x0e9e, 0, 4, (char *)&cc, NULL, 1, NULL);

    cc = 0x18;
    lspmac_send_command( VR_UPLOAD, VR_PMAC_SETMEM
        , 0x0e9e, 0, 4, (char *)&cc, NULL, 1, NULL);
}

lspmac_SockSendDLine( NULL, "I5=0");
lspmac_SockSendDLine( NULL, "ENABLE PLCC 0,2");
lspmac_SockSendDLine( NULL, "DISABLE PLCC 1");
lspmac_SockSendDLine( NULL, "I5=3");

}

```

#### 7.12.4.40 int lspmac\_jogabs\_queue( **lspmac\_motor\_t** \*, double )

Use jog to move motor to requested position.

Definition at line 3483 of file **lspmac.c**.

```

{
return lspmac_move_or_jog_abs_queue( mp,
    requested_position, 1);
}
```

#### 7.12.4.41 int lspmac\_move\_or\_jog\_abs\_queue( **lspmac\_motor\_t** \* mp, double requested\_position, int use\_jo )

Move method for normal stepper and servo motor objects Returns non-zero on abort, zero if OK.

< format string for coordinate system move

< coordinate system bit

< the requested position in units of "counts"  
 < motor and coordinate system;  
 < our axis

#### Parameters

in	<i>mp</i>	The motor to move
in	<i>requested_position</i>	Where to move it
in	<i>use_jo</i>	1 to force jog, 0 for motion prog

Definition at line 3235 of file lsmpmac.c.

```
{
char *fmt;
int q100;
int requested_pos_cnts;
int coord_num, motor_num;
char *axis;
double u2c;
double neutral_pos;
double min_pos, max_pos;
int pos_limit_hit, neg_limit_hit, in_position_band;
struct timespec timeout, now;
int err;

pthread_mutex_lock( &(mp->mutex));

u2c      = lsredis_getd( mp->u2c);
motor_num = lsredis_getl( mp->motor_num);
coord_num = lsredis_getl( mp->coord_num);
axis     = lsredis_getstr( mp->axis);
neutral_pos = lsredis_getd( mp->neutral_pos);
min_pos   = lsredis_getd( mp->min_pos) -
            neutral_pos;
max_pos   = lsredis_getd( mp->max_pos) -
            neutral_pos;
pos_limit_hit = lsredis_getd( mp->pos_limit_hit
    );
neg_limit_hit = lsredis_getd( mp->neg_limit_hit
    );
in_position_band = lsredis_getl( mp->in_position_band
    );

if( u2c == 0.0 || requested_position < min_pos || requested_position >
    max_pos) {
    //
    // Shouldn't try moving a motor that's in trouble
    //
    pthread_mutex_unlock( &(mp->mutex));
    lslogging_log_message( "lsmpmac_move_or_jog_abs_queue:
        %s u2c=%f requested position=%f min allowed=%f max allowed=%f", mp->name
        , u2c, requested_position, min_pos, max_pos);
    lsevents_send_event( "%s Move Aborted", mp->name);
    return 1;
}

if( (neg_limit_hit && (requested_position < mp->position)) || (pos_limit_hit
    && (requested_position > mp->position))) {
    pthread_mutex_unlock( &(mp->mutex));
    lslogging_log_message( "lsmpmac_move_or_jog_abs_queue:
        %s Moving wrong way on limit: requested position=%f current position=%f low
        limit=%d high limit=%d",
        mp->name, requested_position, mp->position
        , neg_limit_hit, pos_limit_hit);
    lsevents_send_event( "%s Move Aborted", mp->name);
    return 2;
}

mp->requested_position = requested_position;
if( mp->nlut > 0 && mp->lut != NULL) {
    mp->requested_pos_cnts = lsmpmac_lut( mp->nlut
        , mp->lut, requested_position);
} else {
    mp->requested_pos_cnts = u2c * (requested_position +
        neutral_pos);
}
requested_pos_cnts = mp->requested_pos_cnts;
```

```

// Bluff if we are already there
//
if( (abs( requested_pos_cnts - mp->actual_pos_cnts) * 16 <
      in_position_band) || (lsredis_getb( mp->active) != 1)) {
    //
    // Lie and say we moved even though we didn't. Who will know? We are
    // within the deadband or not active.
    //
    mp->not_done      = 1;
    mp->motion_seen   = 0;
    mp->command_sent  = 0;

    lsevents_send_event( "%s Moving", mp->name);

    mp->not_done      = 0;
    mp->motion_seen   = 1;
    mp->command_sent  = 1;
}

if( lsredis_getb( mp->active) != 1) {
    //
    // fake the motion for simulated motors
    //
    mp->position = requested_position;
    mp->actual_pos_cnts = requested_pos_cnts;
}
pthread_mutex_unlock( &(mp->mutex));

lsevents_send_event( "%s In Position", mp->name);
return 0;
}

mp->not_done      = 1;
mp->motion_seen   = 0;
mp->command_sent  = 0;

if( use_jog || axis == NULL || *axis == 0) {
    use_jog = 1;
} else {
    use_jog = 0;
    q100 = 1 << (coord_num -1);
}

pthread_mutex_unlock( &(mp->mutex));

if( !use_jog) {
    //
    // Make sure the coordinate system is not moving something, wait if it is
    //
    pthread_mutex_lock( &lspmac_moving_mutex);

    clock_gettime( CLOCK_REALTIME, &now);
    //
    // TODO: Have all moves estimate how long they'll take and use that here
    //
    timeout.tv_sec = now.tv_sec + 60.0;           // a long timeout, but
    // we might really be moving something that takes this long (or longer)
    timeout.tv_nsec = now.tv_nsec;

    err = 0;
    while( err == 0 && (lspmac_moving_flags & q100) != 0)
        err = pthread_cond_timedwait( &lspmac_moving_cond, &
                                      lspmac_moving_mutex, &timeout);

    pthread_mutex_unlock( &lspmac_moving_mutex);

    if( err == ETIMEDOUT) {
        lslogging_log_message( "
        lspmac_move_or_jog_abs_queue: Timed Out. lspmac_moving_flags = %0x", lspmac_moving_flags
        );
        lsevents_send_event( "%s Move Aborted", mp->name);
        return 1;
    }

    //
    // Set the "we are moving this coordinate system" flag
    //
    lspmac_SockSendDPLine( NULL, "M5075=(M5075 | %d)",
                           q100);

    switch( *axis) {
    case 'A':
        fmt = "%d Q16=%d Q100=%d B146R";

```

```

        break;

    case 'B':
        fmt = "%d Q17=%d Q100=%d B147R";
        break;

    case 'C':
        fmt = "%d Q18=%d Q100=%d B148R";
        break;
    case 'X':
        fmt = "%d Q10=%d Q100=%d B140R";
        break;

    case 'Y':
        fmt = "%d Q11=%d Q100=%d B141R";
        break;

    case 'Z':
        fmt = "%d Q12=%d Q100=%d B142R";
        break;

    case 'U':
        fmt = "%d Q13=%d Q100=%d B143R";
        break;

    case 'V':
        fmt = "%d Q14=%d Q100=%d B144R";
        break;

    case 'W':
        fmt = "%d Q15=%d Q100=%d B145R";
        break;
    }

    //
    // Make sure the flag has been seen
    //

    clock_gettime( CLOCK_REALTIME, &now);
    timeout.tv_sec = now.tv_sec + 4.0;           // also a long timeout.
    This should really only take a few milliseconds on a slow day
    timeout.tv_nsec = now.tv_nsec;

    pthread_mutex_lock( &lspmac_moving_mutex);

    err = 0;
    while( err == 0 && (lspmac_moving_flags & q100) == 0)
        err = pthread_cond_timedwait( &lspmac_moving_cond, &
        lspmac_moving_mutex, &timeout);
    pthread_mutex_unlock( &lspmac_moving_mutex);

    if( err == ETIMEDOUT) {
        lslogging_log_message( "
        lspmac_move_or_jog_abs_queue: Did not see flag propagate. Move aborted.");
        lsevents_send_event( "%s Move Aborted", mp->name);
        return 1;
    }
}

pthread_mutex_lock( &(mp->mutex));
if( use_jog) {
    lspmac_SockSendDPLine( mp->name, "#%d j=%d",
    motor_num, requested_pos_cnts);
} else {
    lspmac_SockSendDPLine( mp->name, fmt, coord_num,
    requested_pos_cnts, q100);
}
pthread_mutex_unlock( &(mp->mutex));

free( axis);

return 0;
}

```

#### 7.12.4.42 int lspmac\_move\_or\_jog\_preset\_queue( lspmac\_motor\_t \*, char \*, int )

move using a preset value returns 0 on success, non-zero on error

Definition at line 3444 of file lspmac.c.

```

{
double pos;

```

```

int err;
int rtn;

if( preset == NULL || *preset == 0) {
    lsevents_send_event( "%s Move Aborted", mp->name);
    return 0;
}

err = lsredis_find_preset( mp->name, preset, &pos);

if( err != 0)
    rtn = lspmac_move_or_jog_abs_queue( mp, pos,
        use_jog);
else {
    lsevents_send_event( "%s Move Aborted", mp->name);
    rtn = 1;
}
return rtn;
}

```

#### 7.12.4.43 void lspmac\_move\_or\_jog\_queue( **Ispmac\_motor\_t** \*, double , int )

#### 7.12.4.44 int lspmac\_move\_preset\_queue( **Ispmac\_motor\_t** \* *mp*, char \* *preset\_name* )

Move a given motor to one of its preset positions.

No movement if the preset is not found.

#### Parameters

<i>mp</i>	Ispmac motor pointer
<i>preset_name</i>	Name of the preset to use

Definition at line 2473 of file Ispmac.c.

```

{
double pos;
int err;

lslogging_log_message( "lspmac_move_preset_queue: Called
    with motor %s and preset named '%s'", mp->name, preset_name);

err = lsredis_find_preset( mp->name, preset_name, &pos
    );
if( err == 0)
    return 1;

err = mp->jogAbs( mp, pos);
if( !err)
    lslogging_log_message( "lspmac_move_preset_queue:
        moving %s to preset '%s' (%f)", mp->name, preset_name, pos);
// 
// the abort event should have been sent in moveAbs
//
return err;
}
```

#### 7.12.4.45 int lspmac\_moveabs\_queue( **Ispmac\_motor\_t** \*, double )

Use coordinate system motion program, if available, to move motor to requested position.

Definition at line 3472 of file Ispmac.c.

```

{
return lspmac_move_or_jog_abs_queue( mp,
    requested_position, 0);
}
```

#### 7.12.4.46 int lsmpmac\_moveabs\_wait( lsmpmac\_motor\_t \* mp, double timeout\_secs )

Wait for motor to finish moving.

Assume motion already queued, now just wait

##### Parameters

<i>mp</i>	The motor object to wait for
<i>timeout_secs</i>	The number of seconds to wait for. Fractional values fine.

Definition at line 3498 of file lsmpmac.c.

```

{
    struct timespec timeout, now;
    double isecs, fsecs;
    int err;

    //
    // Copy the queue item for the most recent move request
    //
    clock_gettime( CLOCK_REALTIME, &now);

    fsecs = modf( timeout_secs, &isecs);

    timeout.tv_sec  = now.tv_sec + (long)floor( isecs);
    timeout.tv_nsec = now.tv_nsec + (long)floor( fsecs * 1.0e9);

    timeout.tv_sec  += timeout.tv_nsec / 1000000000;
    timeout.tv_nsec %= 1000000000;

    err = 0;
    pthread_mutex_lock( &(mp->mutex));

    while( err == 0 && mp->command_sent == 0)
        err = pthread_cond_timedwait( &mp->cond, &mp->mutex, &timeout);
    pthread_mutex_unlock( &(mp->mutex));
    if( err != 0) {
        if( err != ETIMEDOUT) {
            lslogging_log_message( "lsmpmac_moveabs_wait:
                unexpected error from timedwait %d tv_sec %ld    tv_nsec %ld", err, timeout.tv_sec,
                timeout.tv_nsec);
        }
        return 1;
    }

    //
    // wait for the motion to have started
    // This will time out if the motion ends before we can read the status back
    // hence the added complication of time stamp of the sent packet.

    err = 0;
    pthread_mutex_lock( &(mp->mutex));
    while( err == 0 && mp->motion_seen == 0)
        err = pthread_cond_timedwait( &(mp->cond), &(mp->mutex), &timeout)
        ;

    if( err != 0) {
        if( err != ETIMEDOUT) {
            lslogging_log_message( "lsmpmac_moveabs_wait:
                unexpected error from timedwait: %d tv_sec %ld    tv_nsec %ld", err, timeout.tv_sec,
                timeout.tv_nsec);
        }
        pthread_mutex_unlock( &(mp->mutex));
        return 1;
    }

    //
    // wait for the motion that we know has started to finish
    //
    err = 0;
    while( err == 0 && mp->not_done)
        err = pthread_cond_timedwait( &(mp->cond), &(mp->mutex), &timeout)
        ;

    if( err != 0) {
        if( err != ETIMEDOUT) {
            lslogging_log_message( "lsmpmac_moveabs_wait:
                unexpected error from timedwait: %d tv_sec %ld    tv_nsec %ld", err, timeout.tv_sec,
                timeout.tv_nsec);
        }
    }
}

```

```

    pthread_mutex_unlock( &(mp->mutex));
    return 1;
}

//
// if return code was not 0 then we know we shouldn't wait for not_done flag.
// In this case the motion ended before we read the status registers
//
pthread_mutex_unlock( &(mp->mutex));
return 0;
}

```

#### 7.12.4.47 void lspmacc\_run( )

Start up the lspmacc thread.

Definition at line 4359 of file lspmacc.c.

```

{
static int first_time = 1;
char **inits;
lspmacc_motor_t *mp;
char evts[64];
int i;
int active;
int motor_num;

pthread_create( &pmac_thread, NULL, lspmacc_worker,
NULL);

if( first_time) {
    first_time = 0;
    lsevents_add_listener( "^CryoSwitchChanged$",
    lspmacc_cryoSwitchChanged_cb);
    lsevents_add_listener( "^scint In Position$",
    lspmacc_scint_maybe_turn_on_dryer_cb);
    lsevents_add_listener( "^scint Moving$",
    lspmacc_scint_maybe_turn_off_dryer_cb);
    lsevents_add_listener( "^scintDried$",
    lspmacc_scint_dried_cb);
    lsevents_add_listener( "^backLight 1$",
    lspmacc_backLight_up_cb);
    lsevents_add_listener( "^backLight 0$",
    lspmacc_backLight_down_cb);
    lsevents_add_listener( "^cam.zoom Moving$",
    lspmacc_light_zoom_cb);
    // lsevents_add_listener( "^Quitting Program$",
    lspmacc_quitting_cb);
    lsevents_add_listener( "^Control-[BCFGV] accepted$",
    lspmacc_request_control_response_cb);
    lsevents_add_listener( "^Full Card Reset$",
    lspmacc_full_card_reset_cb);

    if( pgpmac_use_automotors) {
        lsevents_add_listener( "^scint In Position$",
        lspmacc_scint_maybe_return_sample_cb);
        lsevents_add_listener( "^scint Moving$",
        lspmacc_scint_maybe_move_sample_cb);
    }

    for( i=0; i<lspmacc_nmotors; i++) {
        snprintf( evts, sizeof( evts)-1, "^%s command accepted$",
        lspmacc_motors[i].name);
        evts[sizeof(evts)-1] = 0;
        lsevents_add_listener( evts, lspmacc_command_done_cb
    );
    }
}

lspmacc_zoom_lut_setup();
lspmacc_flight_lut_setup();
lspmacc_blight_lut_setup();
lspmacc_fscint_lut_setup();
}

//
// Clear the command interfaces
//
// lspmacc_SockSendControlCharPrint( "Control-X", '\x18'); // why does this
// kill the initialization?

```

```

{
    uint32_t cc;
    cc = 0;
    lspmacc_send_command( VR_UPLOAD, VR_PMAC_SETMEM
        , 0x0e9e, 0, 4, (char *)&cc, NULL, 1, NULL);

    cc = 0x18;
    lspmacc_send_command( VR_UPLOAD, VR_PMAC_SETMEM
        , 0x0e9e, 0, 4, (char *)&cc, NULL, 1, NULL);
}

// Initialize the MD2 pmac (ie, turn on the right plcc's etc)
// 
/*
for( inits = lsredis_get_string_array(lspmacc_md2_init); *inits != NULL;
    inits++) {
    lspmacc_SockSendDPLine( NULL, *inits);
}

// Initialize the pmac's support for each motor
// (ie, set the various flag for when a motor is active or not)
//
for( i=0; i<lspmacc_nmotors; i++) {
    mp          = &(lspmacc_motors[i]);
    active      = lsredis_getb( mp->active);
    motor_num   = lsredis_getl( mp->motor_num);

    if( motor_num >= 1 && motor_num <= 32) {

        //
        // Set the PMAC to be consistant with redis
        //
        lspmacc_SockSendDPLine( NULL, "I%d16=%f I%d17=%f
            I%d28=%d", motor_num, lsredis_getd( mp->max_speed), motor_num,
            lsredis_getd( mp->max_accel), motor_num, lsredis_getl
            ( mp->in_position_band));
    }

    // if there is a problem with "active" then don't do anything
    // On the other hand, various combinations of yes/no true/false 1/0 should
    // work
    //
    switch( active) {
        case 1:
            inits = lsredis_get_string_array( mp->active_init
            );
            break;

        case 0:
            inits = lsredis_get_string_array( mp->
                inactive_init);
            break;

        default:
            lslogging_log_message( "lspmacc_run: motor %s is
                neither active nor inactive (!?)", mp->name);
            inits = NULL;
    }
    if( inits != NULL) {
        while( *inits != NULL) {
            lspmacc_SockSendDPLine( NULL, *inits);
            inits++;
        }
    }
}
}

```

#### 7.12.4.48 int lspmacc\_set\_motion\_flags ( int \* *mmaskp*, lspmacc\_motor\_t \* *mp\_1*, ... )

Set the coordinate system motion flags (m5075) for the null terminated list of motors that we are planning on running a motion program with.

Note that lspmacc\_est\_move\_time already takes care of this, use when calling a motion program directly

##### Parameters

<i>mmaskp</i>	Returned value of the mask generated. Ignored if null.
<i>mp_1</i>	start of null terminated list of motors.

Definition at line 2743 of file lsmpmac.c.

```

{
va_list arg_ptr;
struct timespec timeout;
int err;
int cn;
int need_flag;
lspmac_motor_t *mp;
int mmask;

mmask = 0;
if( mmaskp != NULL)
    *mmaskp = 0;

if( mp_l==NULL)
    return 0;

// add the coordinate system flags to mmask
// va_start( arg_ptr, mp_l);
for( mp = mp_l; mp!=NULL; mp = va_arg( arg_ptr, lspmac_motor_t
    *)) {
    if( mp->magic != LSPMAC_MAGIC_NUMBER) {
        lslogging_log_message( "lsmpmac_set_motion_flags:
            WARNING: motor list must be NULL terminated. Check your call to
            lsmpmac_set_motion_flags.");
        break;
    }
    cn = lsredis_getl( mp->coord_num);
    if( cn < 1 || cn > 16)
        continue;

    mmask |= 1 << (cn - 1);
}
va_end( arg_ptr);

if( mmaskp != NULL)
    *mmaskp = mmask;

// It could be the flag is already what we want. We might set up a race
// condition if we
// try to set it again. (so don't)
// pthread_mutex_lock( &lspmac_moving_mutex);

if( (lspmac_moving_flags & mmask) != 0)
    need_flag = 0;
else
    need_flag = 1;

pthread_mutex_unlock( &lspmac_moving_mutex);

if( !need_flag)
    return 0;

// Set m5075 and make sure it propagates
// lsmpmac_SockSendDPLine( NULL, "M5075=(M5075 | %d)", mmask
    );
clock_gettime( CLOCK_REALTIME, &timeout);
timeout.tv_sec += 2;

err = 0;
pthread_mutex_lock( &lspmac_moving_mutex);
while( err == 0 && (lspmac_moving_flags & mmask) != mmask)
    err = pthread_cond_timedwait( &lspmac_moving_cond, &
        lspmac_moving_mutex, &timeout);

pthread_mutex_unlock( &lspmac_moving_mutex);

if( err == ETIMEDOUT) {
    lslogging_log_message( "lsmpmac_set_motion_flags: timed
        out waiting for motion %d flag to be set", mmask);
    return 1;
}
return 0;
}

```

#### 7.12.4.49 pmac\_cmd\_queue\_t\* lspmacc\_SockSendControlCharPrint ( char \* event, char c )

Send a control character.

##### Parameters

in	<i>event</i>	base name for events
	<i>c</i>	The control character to send

Definition at line 1135 of file lspmacc.c.

```

    {
lspmacc_control_char = c;
//  return lspmacc_send_command( VR_DOWNLOAD, VR_PMAC_SENDCTRLCHAR, c, 0,
//                               1400, NULL, lspmacc_SendControlReplyPrintCB, 0, event);
return lspmacc_send_command( VR_UPLOAD,
                           VR_CTRL_RESPONSE, c, 0, 1400, NULL,
                           lspmacc_SendControlReplyPrintCB, 0, event);
}

```

#### 7.12.4.50 void lspmacc\_SockSendDPCControlChar ( char \* event, char c )

use dpram ascii interface to send a control character

Definition at line 2067 of file lspmacc.c.

```

{
uint16_t buff;

buff = 0x07 & c;
lspmacc_send_command( VR_UPLOAD, VR_PMAC_SETMEM
, 0x0e9e, 0, 2, (char *)&buff, lspmacc_SockSendDPControlCharCB
, 1, event);
if( event != NULL)
lsevents_send_event( "%s queued", event);
}

```

#### 7.12.4.51 void lspmacc\_SockSendDPLine ( char \*, char \* fmt, ... )

prepare (queue up) a line to send the dpram ascii command interface

Definition at line 2024 of file lspmacc.c.

```

{
va_list arg_ptr;
uint32_t index;
char *pl;

pthread_mutex_lock( &lspmacc_ascii_mutex);
index = lspmacc_dpascii_on++ % LSPMAC_DPASCII_QUEUE_LENGTH
;

pl = lspmacc_dpascii_queue[index].pl;
va_start( arg_ptr, fmt);
vsnprintf( pl, 159, fmt, arg_ptr);
pl[159] = 0;
va_end( arg_ptr);

lspmacc_dpascii_queue[index].event = event;
pthread_mutex_unlock( &lspmacc_ascii_mutex);
}

```

#### 7.12.4.52 pmac\_cmd\_queue\_t\* lspmacc\_SockSendline ( char \* event, char \* fmt, ... )

Send a one line command.

Uses printf style arguments.

**Parameters**

in	<i>event</i>	base name for events
in	<i>fmt</i>	Printf style format string

Definition at line 1092 of file lsmpmac.c.

```

{
va_list arg_ptr;
char payload[1400];

va_start( arg_ptr, fmt);
vsnprintf( payload, sizeof(payload)-1, fmt, arg_ptr);
payload[ sizeof(payload)-1 ] = 0;
va_end( arg_ptr);

lslogging_log_message( "%s", payload);

return lsmpmac_send_command( VR_DOWNLOAD,
    VR_PMAC_SNDLINE, 0, 0, strlen( payload), payload,
    lsmpmac_GetShortReplyCB, 0, event);
}
}
```

**7.12.4.53 void lsmpmac\_video\_rotate( double secs )**

Special motion program to collect centering video.

Definition at line 2705 of file lsmpmac.c.

```

{
double q10;           // starting position (counts)
double q11;           // delta counts
double q12;           // milliseconds to run over delta

double u2c;
double neutral_pos;

if( secs <= 0.0)
    return;

omega_zero_search = 1;

pthread_mutex_lock( &(omega->mutex));
u2c      = lsredis_getd( omega->u2c);
neutral_pos = lsredis_getd( omega->neutral_pos);

q10 = neutral_pos * u2c;
q11 = 360.0 * u2c;
q12 = 1000 * secs;

omega_zero_velocity = 360.0 * u2c / secs; // counts/second to back calculate zero crossing time

lsmpmac_SockSendDPLine( omega->name, "&1
    Q10=%1f Q11=%1f Q12=%1f Q13=(I117) Q14=(I116) B240R", q10, q11, q12);
pthread_mutex_unlock( &(omega->mutex));
}
}
```

**7.12.4.54 int lsredis\_cmpnstr( lsredis\_obj\_t \* p, char \* s, int n )**

Definition at line 256 of file lsredis.c.

```

{
int rtn;
pthread_mutex_lock( &p->mutex);
while( p->valid == 0)
    pthread_cond_wait( &p->cond, &p->mutex);

rtn = strncmp( p->value, s, n);
pthread_mutex_unlock( &p->mutex);
return rtn;
}
```

#### 7.12.4.55 int lsredis\_cmpstr( lsredis\_obj\_t \* p, char \* s )

Definition at line 245 of file lsredis.c.

```

{
int rtn;
pthread_mutex_lock( &p->mutex );
while( p->valid == 0 )
    pthread_cond_wait( &p->cond, &p->mutex );

rtn = strcmp( p->value, s );
pthread_mutex_unlock( &p->mutex );
return rtn;
}

```

#### 7.12.4.56 void lsredis\_config( )

Definition at line 1097 of file lsredis.c.

```

{
char hostname[128], lhostname[128];
int i;

pthread_mutexattr_init( &mutex_initializer );
pthread_mutexattr_settype( &mutex_initializer,
    PTHREAD_MUTEX_RECURSIVE );

if( gethostname( hostname, sizeof(hostname)-1) ) {
    lslogging_log_message( "lsredis_init: cannot get our
        own host name. Cannot configure redis variables." );
} else {
    for( i=0; i<strlen(hostname); i++ ) {
        lhostname[i] = tolower( hostname[i] );
    }
    lhostname[i] = 0;

    lslogging_log_message( "lsredis_init: our host name is
        '%s', lhostname";
    redisAsyncCommand( roac, lsredis_configCB, NULL, "
        hgetall config.%s", lhostname );
}

pthread_mutex_lock( &lsredis_config_mutex );
while( lsredis_head == NULL )
    pthread_cond_wait( &lsredis_config_cond, &lsredis_config_mutex );
pthread_mutex_unlock( &lsredis_config_mutex );
}

```

#### 7.12.4.57 int lsredis\_find\_preset( char \* base, char \* preset\_name, double \* dval )

Get the value of the given preset and return it in dval Returns 0 on error, non-zero on success;.

Definition at line 903 of file lsredis.c.

```

{
char s[512];
int err;
ENTRY entry_in, *entry_outp;
lsredis_preset_list_t *pl;

snprintf( s, sizeof( s)-1, "%s%s", motor_name, preset_name );
s[sizeof(s)-1] = 0;

entry_in.key = s;
entry_in.data = NULL;
err = hsearch_r( entry_in, FIND, &entry_outp, &lsredis_preset_ht
    );
if( err == 0 ) {
    // not found (or some other problem that means we don't have an answer
    //
    // Maybe someone added a new preset and we don't know about it yet
    //
}

```

```

lsredis_load_presets( motor_name);
err = hsearch_r( entry_in, FIND, &entry_outp, &lsredis_preset_ht
);
if( err == 0) {
//
// Guess not. Give up. We tried
//
*dval = 0.0;
return 0;
}
pl = entry_outp->data;
*dval = lsredis_getd( pl->position);
return 1;
}

```

#### 7.12.4.58 int lsredis\_find\_preset\_index\_by\_position ( lsmac\_motor\_t \* mp )

For the given motor object return the index of the current preset or -1 if we are not at a preset position.

Definition at line 985 of file lsredis.c.

```

{
lsredis_obj_t *p;
int plength;
int i;
double ur, pos;

p = lsredis_get_obj( "%s.presets.length", mp->name);
plength = lsredis_get_or_set_1( p, 0);

if( plength <= 0) {
    return -1;
}

ur = lsredis_getd( mp->update_resolution);
pos = lsmac_getPosition( mp);

for( i=0; i<plength; i++) {
    p = lsredis_get_obj( "%s.presets.%d.position", mp->name,
        i);
    if( fabs( pos - lsredis_getd( p)) <= ur) {
        return i;
    }
}
return -1;
}

```

#### 7.12.4.59 lsredis\_obj\_t\* lsredis\_get\_obj ( char \* , ... )

Definition at line 596 of file lsredis.c.

```

{
lsredis_obj_t *rtn;
va_list arg_ptr;
char k[512];
char *kp;
int nkp;

va_start( arg_ptr, fmt);
vsnprintf( k, sizeof(k)-1, fmt, arg_ptr);
k[sizeof(k)-1] = 0;
va_end( arg_ptr);

nkp = strlen(k) + strlen( lsredis_head) + 16; // 16
    is overkill. I know. Get over it.
kp = calloc( nkp, sizeof( char));
if( kp == NULL) {
    lslogging_log_message( "lsredis_get_obj: Out of memory
    ");
    exit( -1);
}

snprintf( kp, nkp-1, "%s.%s", lsredis_head, k);
kp[nkp-1] = 0;

pthread_mutex_lock( &lsredis_mutex);

```

```

while( lsredis_running == 0)
    pthread_cond_wait( &lsredis_cond, &lsredis_mutex);

rtn = _lsredis_get_obj( kp);
pthread_mutex_unlock( &lsredis_mutex);

free( kp);
return rtn;
}

```

#### 7.12.4.60 char\*\* lsredis\_get\_string\_array ( lsredis\_obj\_t \* p )

Definition at line 437 of file lsredis.c.

```

{
char **rtn;

pthread_mutex_lock( &p->mutex);
while( p->valid == 0)
    pthread_cond_wait( &p->cond, &p->mutex);

rtn = p->avalue;
pthread_mutex_unlock( &p->mutex);

return rtn;
}

```

#### 7.12.4.61 int lsredis\_getb ( lsredis\_obj\_t \* p )

Definition at line 450 of file lsredis.c.

```

{
int rtn;

pthread_mutex_lock( &p->mutex);
while( p->valid == 0)
    pthread_cond_wait( &p->cond, &p->mutex);

rtn = p->bvalue;
pthread_mutex_unlock( &p->mutex);

return rtn;
}

```

#### 7.12.4.62 char lsredis\_getc ( lsredis\_obj\_t \* p )

Definition at line 463 of file lsredis.c.

```

{
int rtn;

pthread_mutex_lock( &p->mutex);
while( p->valid == 0)
    pthread_cond_wait( &p->cond, &p->mutex);

rtn = p->cvalue;
pthread_mutex_unlock( &p->mutex);

return rtn;
}

```

#### 7.12.4.63 double lsredis\_getd ( lsredis\_obj\_t \* p )

Definition at line 385 of file lsredis.c.

```

    {
double rtn;

pthread_mutex_lock( &p->mutex );
while( p->valid == 0 )
    pthread_cond_wait( &p->cond, &p->mutex );

rtn = p->dvalue;
pthread_mutex_unlock( &p->mutex );

return rtn;
}

```

#### 7.12.4.64 long int lsredis\_getl ( lsredis\_obj\_t \*p )

Definition at line 398 of file lsredis.c.

```

{
long int rtn;

pthread_mutex_lock( &p->mutex );
while( p->valid == 0 )
    pthread_cond_wait( &p->cond, &p->mutex );

rtn = p->lvalue;
pthread_mutex_unlock( &p->mutex );

return rtn;
}

```

#### 7.12.4.65 char\* lsredis\_getstr ( lsredis\_obj\_t \*p )

return a copy of the key's string value be sure to free the result

Definition at line 284 of file lsredis.c.

```

{
char *rtn;

// 
// Have to use strdup since we cannot guarantee that p->value won't be freed
// while the caller is still using it
//
pthread_mutex_lock( &p->mutex );
while( p->valid == 0 )
    pthread_cond_wait( &p->cond, &p->mutex );

rtn = strdup(p->value);
pthread_mutex_unlock( &p->mutex );
return rtn;
}

```

#### 7.12.4.66 void lsredis\_init ( )

Initialize this module, that is, set up the connections.

##### Parameters

<i>pub</i>	Publish under this (unique) name
<i>re</i>	Regular expression to select keys we want to mirror
<i>head</i>	Prepend this (+ a dot) to the beginning of requested objects

Definition at line 1129 of file lsredis.c.

```

{
int err;

```

```

// set up hash map to store redis objects
//
err = hcreate_r( 8192, &lsredis_htab);
if( err == 0) {
    lslogging_log_message( "lsredis_init: Cannot create
        hash table. Really bad things are going to happen. hcreate_r returned %d", err);
}

pthread_cond_init( &lsredis_cond, NULL);

subac = redisAsyncConnect("127.0.0.1", 6379);
if( subac->err) {
    lslogging_log_message( "Error: %s", subac->errstr
    );
}

subfd.fd      = subac->c.fd;
subfd.events   = 0;
subac->ev.data = &subfd;
subac->ev.addRead = lsredis_addRead;
subac->ev.delRead = lsredis_delRead;
subac->ev.addWrite = lsredis_addWrite;
subac->ev.delWrite = lsredis_delWrite;
subac->ev.cleanup = lsredis_cleanup;

roac = redisAsyncConnect("127.0.0.1", 6379);
if( roac->err) {
    lslogging_log_message( "Error: %s", roac->errstr);
}

rofd.fd      = roac->c.fd;
rofd.events   = 0;
roac->ev.data = &rofd;
roac->ev.addRead = lsredis_addRead;
roac->ev.delRead = lsredis_delRead;
roac->ev.addWrite = lsredis_addWrite;
roac->ev.delWrite = lsredis_delWrite;
roac->ev.cleanup = lsredis_cleanup;

wrac = redisAsyncConnect("127.0.0.1", 6379);
if( wrac->err) {
    lslogging_log_message( "Error: %s", wrac->errstr);
}

wrfd.fd      = wrac->c.fd;
wrfd.events   = 0;
wrac->ev.data = &wrfd;
wrac->ev.addRead = lsredis_addRead;
wrac->ev.delRead = lsredis_delRead;
wrac->ev.addWrite = lsredis_addWrite;
wrac->ev.delWrite = lsredis_delWrite;
wrac->ev.cleanup = lsredis_cleanup;

// separate hash table for the presets
//
hcreate_r( lsredis_preset_max_n * 2, &lsredis_preset_ht
    );

pthread_mutex_init( &lsredis_preset_list_mutex, &
    mutex_initializer);
pthread_mutex_init( &lsredis_config_mutex, &
    mutex_initializer);
pthread_cond_init( &lsredis_config_cond, NULL);
}

```

#### 7.12.4.67 void lsredis\_load\_presets ( char \* *motor\_name* )

update the presets hash table for the named motor

Definition at line 830 of file lsredis.c.

```

{
lsredis_obj_t *p;
lsredis_preset_list_t *pl;
int plength;
char *preset_name;
int i;
int key_length;
ENTRY entry_in, *entry_outp;
```

```

p = lsredis_get_obj( "%s.presets.length", motor_name);
plength = lsredis_get_or_set_l( p, 0);
if( plength <= 0)
    return;

pthread_mutex_lock( &lsredis_preset_list_mutex);

for( i=0; i<plength; i++) {
    pl = calloc( 1, sizeof( lsredis_preset_list_t));
    pl->name      = lsredis_get_obj( "%s.presets.%d.name",
        motor_name, i);
    pl->position   = lsredis_get_obj( "
        %s.presets.%d.position", motor_name, i);
    pl->index      = i;

    preset_name    = lsredis_getstr( pl->name);
    key_length     = strlen( motor_name) + strlen( preset_name) + 1;
    pl->key        = calloc( key_length, 1);

    pl->next       = lsredis_preset_list;
    lsredis_preset_list = pl;

    sprintf( pl->key, key_length, "%s%s", motor_name, preset_name);

    entry_in.key   = pl->key;
    entry_in.data  = pl;
    hsearch_r( entry_in, ENTER, &entry_outp, &lsredis_preset_ht
    );
    if( entry_outp->data != pl) {
        //
        // The key was already there or we couldn't add it
        //
        if( entry_outp->data == NULL)
            lslogging_log_message( "lsredis_load_presets:
                could not add preset '%s' for motor '%s'", preset_name, motor_name);

        free( pl->key);
        free( pl);
    } else {
        //
        // We've successfully added the new key
        //
        lsredis_preset_n++;
        //
        // Resize the hash table if we are starting to fill it up
        // Generally we prefer a sparse table
        //
        if( lsredis_preset_n >= lsredis_preset_max_n
        ) {
            lslogging_log_message( "lsredis_load_presets:
                increasing preset hash table size. max now %d", lsredis_preset_max_n
            );
            hdestroy_r( &lsredis_preset_ht);
            lsredis_preset_max_n *= 2;
            hcreate_r( 2 * lsredis_preset_max_n, &
            lsredis_preset_ht);
            for( pl = lsredis_preset_list; pl != NULL; pl = pl->
            next) {
                entry_in.key   = pl->key;
                entry_in.data  = pl;
                hsearch_r( entry_in, ENTER, &entry_outp, &lsredis_preset_ht
            );
            }
            lslogging_log_message( "lsredis_load_presets: done
                increasing preset hash table size.", lsredis_preset_max_n);
        }
    }
    free( preset_name);
}
pthread_mutex_unlock( &lsredis_preset_list_mutex);
}

```

#### 7.12.4.68 int lsredis\_reexec ( const regex\_t \* preg, lsredis\_obj\_t \* p, size\_t nmatch, regmatch\_t \* pmatch, int eflags )

Definition at line 267 of file lsredis.c.

```

{
int rtn;

pthread_mutex_lock( &p->mutex);
while( p->valid == 0)

```

```

pthread_cond_wait( &p->cond, &p->mutex);

rtn = regexec( preg, p->value, nmatch, pmatch, eflags);

pthread_mutex_unlock( &p->mutex);

return rtn;
}

```

#### 7.12.4.69 void lsredis\_run( )

Definition at line 1319 of file lsredis.c.

```

{
pthread_create( &lsredis_thread, NULL, lsredis_worker
               , NULL);
}

```

#### 7.12.4.70 void lsredis\_set\_preset( char \* base, char \* preset\_name, double dval )

set the given preset to the given value create a new preset if we can't find it

Definition at line 940 of file lsredis.c.

```

{
char s[512];
int plength;
int err;
ENTRY entry_in, *entry_outp;
lsredis_obj_t *p, *presets_length_p;
lsredis_preset_list_t *pl;

snprintf( s, sizeof( s)-1, "%s%s", motor_name, preset_name);
s[sizeof(s)-1] = 0;

entry_in.key = s;
entry_in.data = NULL;
err = hsearch_r( entry_in, FIND, &entry_outp, &lsredis_preset_ht
                 );
if( err != 0) {
    //
    // Found it. Things are simple.
    //
    pl = entry_outp->data;
    lsredis_setstr( pl->position, "%.3f", dval);
    return;
}
//
// OK, our preset was not found, add it
//
presets_length_p = lsredis_get_obj( "%s.presets.length",
                                    motor_name);
plength = lsredis_get_or_set_1( presets_length_p, 0);
plength += 1;

snprintf( s, sizeof( s)-1, "%s.%s.presets.%d.name", lsredis_head,
          motor_name, plength-1);
s[sizeof(s)-1] = 0;

p = lsredis_get_obj( "%s.presets.%d.name", motor_name, plength
                     -1);
lsredis_setstr( p, "%s", preset_name);

p = lsredis_get_obj( "%s.presets.%d.position", motor_name,
                     plength-1);
lsredis_setstr( p, "%.3f", dval);

lsredis_setstr( presets_length_p, "%ld", plength);

lsredis_load_presets( motor_name);
}

```

## 7.12.4.71 void lsredis\_setstr ( lsredis\_obj\_t \* p, char \* fmt, ... )

Set the value and update redis.

Note that lsredis\_set\_value sets the value based on redis while here we set redis based on the value Arbitray maximum string length set here. TODO: Probably this limit should be removed at some point.

redisAsyncCommandArgv used instead of redisAsyncCommand 'cause it's easier (and possible) to deal with strings that would otherwise cause hiredis to emit a bad command, like those containing spaces. < up the count of times we need to see ourselves published before we start listening to others again

< Unlock to prevent deadlock in case the service routine needs to set our value

< redisAsyncCommandArgv shouldn't need to access this after it's made up it's packet (before it returns) so we should be OK with this location disappearing soon.

Definition at line 309 of file lsredis.c.

```

va_list arg_ptr;
char v[512];
char *argv[4];

va_start( arg_ptr, fmt);
vsnprintf( v, sizeof(v)-1, fmt, arg_ptr);
v[sizeof(v)-1] = 0;
va_end( arg_ptr);

pthread_mutex_lock( &p->mutex);

// 
// Don't send an update if a good value has not changed
//
if( p->valid && strcmp( v, p->value) == 0) {
    // nothing to do
    pthread_mutex_unlock( &p->mutex);
    return;
}

p->wait_for_me++;
pthread_mutex_unlock( &p->mutex);

argv[0] = "HSET";
argv[1] = p->key;
argv[2] = "VALUE";
argv[3] = v;

pthread_mutex_lock( &lsredis_mutex);
while( lsredis_running == 0)
    pthread_cond_wait( &lsredis_cond, &lsredis_mutex);

redisAsyncCommand( wrac, NULL, NULL, "MULTI");
redisAsyncCommandArgv( wrac, NULL, NULL, 4, (const char **)argv, NULL);

redisAsyncCommand( wrac, NULL, NULL, "PUBLISH %s %s", lsredis_publisher
    , p->key);
redisAsyncCommand( wrac, NULL, NULL, "EXEC");
pthread_mutex_unlock( &lsredis_mutex);

// Assume redis will take exactly the value we sent it
//
pthread_mutex_lock( &p->mutex);
lsredis_set_value( p, v);
pthread_cond_signal( &p->cond);
pthread_mutex_unlock( &p->mutex);
}

```

## 7.12.4.72 void ltest\_main ( )

Definition at line 92 of file ltest.c.

```

{
ltest_lspmac_est_move_time();
}
```

#### 7.12.4.73 void lstimber\_init( )

Initialize the timer list and pthread stuff.

Definition at line 270 of file lstimber.c.

```

    {
int i;

for( i=0; i<LSTIMER_LIST_LENGTH; i++) {
    lstimer_list[i].shots = 0;
}

pthread_mutex_init( &lstimber_mutex, NULL);
pthread_cond_init( &lstimber_cond, NULL);
}

```

#### 7.12.4.74 void lstimber\_run( )

Start up our thread.

Definition at line 284 of file lstimber.c.

```

{
pthread_create( &lstimber_thread, NULL, lstimber_worker
               , NULL);
}

```

#### 7.12.4.75 void lstimber\_set\_timer( char \* event, int shots, unsigned long int secs, unsigned long int nsecs )

Create a timer.

##### Parameters

<i>event</i>	Name of the event to send when the timer goes off
<i>shots</i>	Number of times to run. 0 means never, -1 means forever
<i>secs</i>	Number of seconds to wait
<i>nsecs</i>	Number of nano-seconds to run in addition to secs

Definition at line 63 of file lstimber.c.

```

    {
int i;
struct timespec now;

// Time we were called. Delay is based on call time, not queued time
// clock_gettime( CLOCK_REALTIME, &now);

// Make sure our event is registered (saves a tiny bit of time later)
// lsevents_preregister_event( event);

pthread_mutex_lock( &lstimber_mutex);

for( i=0; i<LSTIMER_LIST_LENGTH; i++) {
    if( lstimer_list[i].shots == 0)
        break;
}

if( i == LSTIMER_LIST_LENGTH) {
    pthread_mutex_unlock( &lstimber_mutex);

    lslogging_log_message( "lstimber_set_timer: out of
                           timers for event: %s, shots: %d, secs: %u,
                           nsecs: %u",
                           event, shots, secs, nsecs);
    return;
}

```

```

}

strncpy( lstimer_list[i].event, event, LSEVENTS_EVENT_LENGTH
        - 1);
lstimer_list[i].event[LSEVENTS_EVENT_LENGTH
        - 1] = 0;
lstimer_list[i].shots      = shots;
lstimer_list[i].delay_secs  = secs;
lstimer_list[i].delay_nsecs = nsecs;

lstimer_list[i].next_secs   = secs + now.tv_sec + (
    now.tv_nsec + nsecs) / 1000000000;
lstimer_list[i].next_nsecs = (now.tv_nsec + nsecs
    ) % 1000000000;
lstimer_list[i].last_secs   = 0;
lstimer_list[i].last_nsecs = 0;

lstimer_list[i].ncalls      = 0;
lstimer_list[i].init_secs   = now.tv_sec;
lstimer_list[i].init_nsecs = now.tv_nsec;

if( shots != 0) {
    lstimer_active_timers++;
    new_timer++;
}

pthread_cond_signal( &lstimer_cond);
pthread_mutex_unlock( &lstimer_mutex);
}

```

#### 7.12.4.76 void lstimer\_unset\_timer( char \* event )

Unsets all timers for the given event.

Definition at line 46 of file lstimer.c.

```

{
int i;

for( i=0; i<LSTIMER_LIST_LENGTH; i++) {
    if( strcmp( event, lstimer_list[i].event) == 0) {
        lstimer_list[i].shots = 0;
    }
}
}

```

#### 7.12.4.77 void lsupdate\_init( )

#### 7.12.4.78 void lsupdate\_run( )

#### 7.12.4.79 void md2cmds\_init( )

Initialize the md2cmds module.

Definition at line 1885 of file md2cmds.c.

```

{
ENTRY hloader, *hrtnval;
int i, err;
int ncmds;

pthread_mutexattr_t mutex_initializer;
pthread_mutexattr_init( &mutex_initializer);
pthread_mutexattr_settype( &mutex_initializer, PTHREAD_MUTEX_RECURSIVE);

pthread_mutex_init( &md2cmds_mutex, &mutex_initializer);
pthread_cond_init( &md2cmds_cond, NULL);

pthread_mutex_init( &md2cmds_moving_mutex, &
    mutex_initializer);
pthread_cond_init( &md2cmds_moving_cond, NULL);

```

```

pthread_mutex_init( &md2cmds_homing_mutex, &
    mutex_initializer);
pthread_cond_init( &md2cmds_homing_cond, NULL);

err = regcomp( &md2cmds_cmd_regex, " *([^\n]+) (([^\n]+)\\\
    .presets\\..)*([^\n]*)([^\n]*", REG_EXTENDED);
if( err != 0) {
    int nerrmsg;
    char *errmsg;

    nerrmsg = regerror( err, &md2cmds_cmd_regex, NULL, 0);
    if( nerrmsg > 0) {
        errmsg = calloc( nerrmsg, sizeof( char));
        nerrmsg = regerror( err, &md2cmds_cmd_regex, errmsg,
        nerrmsg);
        lslogging_log_message( "md2cmds_init: %s", errmsg);
        free( errmsg);
    }
}

md2cmds_md_status_code = lsredis_get_obj
    ( "md2_status_code");
lsredis_setstr( md2cmds_md_status_code, "
    7");

ncmds = sizeof(md2cmds_cmd_kvs)/sizeof(md2cmds_cmd_kvs
    [0]);
if( pgpmac_use_pg) {
    ncmds += sizeof(md2cmds_cmd_pg_kvs)/sizeof(
        md2cmds_cmd_pg_kvs[0]);
}
hcreate_r( 2 * ncmds, &md2cmds_hmap);

for( i=0; i<sizeof(md2cmds_cmd_kvs)/sizeof(md2cmds_cmd_kvs
    [0]); i++) {
    hloader.key   = md2cmds_cmd_kvs[i].k;
    hloader.data = md2cmds_cmd_kvs[i].v;
    err = hsearch_r( hloader, ENTER, &hrtnval, &md2cmds_hmap);
    if( err == 0) {
        lslogging_log_message( "md2cmds_init: hsearch_r
            returned an error for item %d: %s", i, strerror( errno));
    }
}

if( pgpmac_use_pg) {
    for( i=0; i<sizeof(md2cmds_cmd_pg_kvs)/sizeof(
        md2cmds_cmd_pg_kvs[0]); i++) {
        hloader.key   = md2cmds_cmd_pg_kvs[i].k;
        hloader.data = md2cmds_cmd_pg_kvs[i].v;
        err = hsearch_r( hloader, ENTER, &hrtnval, &md2cmds_hmap);
        if( err == 0) {
            lslogging_log_message( "md2cmds_init: hsearch_r
                returned an error for item %d: %s", i, strerror( errno));
        }
    }
}
}

```

#### 7.12.4.80 void md2cmds\_push\_queue ( char \* *action* )

Definition at line 79 of file md2cmds.c.

```

{
if( pthread_mutex_trylock( &md2cmds_mutex) == 0) {
    strncpy( md2cmds_cmd, action, MD2CMDS_CMD_LENGTH
        -1);
    md2cmds_cmd[MD2CMDS_CMD_LENGTH-1] = 0;
    pthread_cond_signal( &md2cmds_cond);
    pthread_mutex_unlock( &md2cmds_mutex);
} else {
    lslogging_log_message( "md2cmds_push_queue: MD2
        command '%s' ignored. Already running '%s'", action, md2cmds_cmd);
}
}
```

## 7.12.4.81 void md2cmds.run( )

Start up the thread.

Definition at line 1952 of file md2cmds.c.

```

{
pthread_create( &md2cmds_thread, NULL,
    md2cmds_worker, NULL);
lsevents_add_listener( "^omega crossed zero$",
    md2cmds_rotate_cb);
lsevents_add_listener( "^omega In Position$",
    md2cmds_maybe_rotate_done_cb);
lsevents_add_listener( ".+ (Moving|In Position)$",
    md2cmds_maybe_done_moving_cb);
lsevents_add_listener( "(.+) (Homing|Homed)$",
    md2cmds_maybe_done_homing_cb);
lsevents_add_listener( "^capz (Moving|In Position)$",
    md2cmds_time_capz_cb);
lsevents_add_listener( "^Coordsys 1 Stopped$",
    md2cmds_coordsys_1_stopped_cb);
lsevents_add_listener( "^Coordsys 2 Stopped$",
    md2cmds_coordsys_2_stopped_cb);
lsevents_add_listener( "^Coordsys 3 Stopped$",
    md2cmds_coordsys_3_stopped_cb);
lsevents_add_listener( "^Coordsys 4 Stopped$",
    md2cmds_coordsys_4_stopped_cb);
lsevents_add_listener( "^Coordsys 5 Stopped$",
    md2cmds_coordsys_5_stopped_cb);
lsevents_add_listener( "^Coordsys 7 Stopped$",
    md2cmds_coordsys_7_stopped_cb);
lsevents_add_listener( "^cam.zoom Moving$",
    md2cmds_set_scale_cb);
}

```

## 7.12.4.82 void pgpmac\_printf( char \* fmt, ... )

Terminal output routine ala printf.

## Parameters

in	fmt	Printf style formating string
----	-----	-------------------------------

Definition at line 443 of file pgpmac.c.

```

{
va_list arg_ptr;

pthread_mutex_lock( &ncurses_mutex);

va_start( arg_ptr, fmt);
vwprintw( term_output, fmt, arg_ptr);
va_end( arg_ptr);

wnoutrefresh( term_output);
doupdate();
pthread_mutex_unlock( &ncurses_mutex);

}

```

## 7.12.4.83 void pgpmac\_request\_stay\_of\_execution( int secs )

Postpone the day of reckoning This assumes the quit\_cb routine is called once a second.

Definition at line 464 of file pgpmac.c.

```

{
pthread_mutex_lock( &doomsday_mutex);
if( secs > doomsday_count)
    doomsday_count = secs;
pthread_mutex_unlock( &doomsday_mutex);
}

```

7.12.4.84 void PmacSockSendline ( char \* s )

## 7.12.5 Variable Documentation

7.12.5.1 **lspmac\_motor\_t\* alignx**

Alignment stage X.

Definition at line 103 of file lspmac.c.

7.12.5.2 **lspmac\_motor\_t\* aligny**

Alignment stage Y.

Definition at line 104 of file lspmac.c.

7.12.5.3 **lspmac\_motor\_t\* alignz**

Alignment stage X.

Definition at line 105 of file lspmac.c.

7.12.5.4 **lspmac\_motor\_t\* anal**

Polaroid analyzer motor.

Definition at line 106 of file lspmac.c.

7.12.5.5 **lspmac\_motor\_t\* apery**

Aperture Y.

Definition at line 108 of file lspmac.c.

7.12.5.6 **lspmac\_motor\_t\* aperz**

Aperture Z.

Definition at line 109 of file lspmac.c.

7.12.5.7 **lspmac\_bi\_t\* arm\_parked**

(whose arm? parked where?)

Definition at line 146 of file lspmac.c.

7.12.5.8 **lspmac\_motor\_t\* blight**

Back Light DAC.

Definition at line 120 of file lspmac.c.

7.12.5.9 **lspmac\_bi\_t\* blight\_down**

Backlight is down.

Definition at line 136 of file lspmac.c.

**7.12.5.10 Ispmac\_motor\_t\* blight\_f**

Back light scale factor.

Definition at line 129 of file Ispmac.c.

**7.12.5.11 Ispmac\_motor\_t\* blight\_ud**

Back light Up/Down actuator.

Definition at line 124 of file Ispmac.c.

**7.12.5.12 Ispmac\_bi\_t\* blight\_up**

Backlight is up.

Definition at line 137 of file Ispmac.c.

**7.12.5.13 Ispmac\_motor\_t\* capy**

Capillary Y.

Definition at line 110 of file Ispmac.c.

**7.12.5.14 Ispmac\_motor\_t\* capz**

Capillary Z.

Definition at line 111 of file Ispmac.c.

**7.12.5.15 Ispmac\_motor\_t\* cenx**

Centering Table X.

Definition at line 113 of file Ispmac.c.

**7.12.5.16 Ispmac\_motor\_t\* ceny**

Centering Table Y.

Definition at line 114 of file Ispmac.c.

**7.12.5.17 Ispmac\_motor\_t\* cryo**

Move the cryostream towards or away from the crystal.

Definition at line 125 of file Ispmac.c.

**7.12.5.18 Ispmac\_bi\_t\* cryo\_back**

cryo is in the back position

Definition at line 138 of file Ispmac.c.

**7.12.5.19 `Ispmac.bi_t* cryo_switch`**

that little toggle switch for the cryo

Definition at line 135 of file Ispmac.c.

**7.12.5.20 `Ispmac.motor_t* dryer`**

blow air on the scintilator to dry it off

Definition at line 126 of file Ispmac.c.

**7.12.5.21 `Ispmac.bi_t* etel_init_ok`**

ETEL initialized OK.

Definition at line 143 of file Ispmac.c.

**7.12.5.22 `Ispmac.bi_t* etel_on`**

ETEL is on.

Definition at line 142 of file Ispmac.c.

**7.12.5.23 `Ispmac.bi_t* etel_ready`**

ETEL is ready.

Definition at line 141 of file Ispmac.c.

**7.12.5.24 `Ispmac.motor_t* flight`**

Front Light DAC.

Definition at line 119 of file Ispmac.c.

**7.12.5.25 `Ispmac.motor_t* flight_f`**

Front light scale factor.

Definition at line 130 of file Ispmac.c.

**7.12.5.26 `Ispmac.motor_t* flight_oo`**

Turn front light on/off.

Definition at line 128 of file Ispmac.c.

**7.12.5.27 `Ispmac.motor_t* fluo`**

Move the fluorescence detector in/out.

Definition at line 127 of file Ispmac.c.

**7.12.5.28 Ispmac.bi\_t\* fluor\_back**

fluor is in the back position

Definition at line 139 of file Ispmac.c.

**7.12.5.29 Ispmac\_motor\_t\* fscint**

Scintillator Piezo DAC.

Definition at line 121 of file Ispmac.c.

**7.12.5.30 Ispmac\_motor\_t\* fshut**

Fast shutter.

Definition at line 118 of file Ispmac.c.

**7.12.5.31 Ispmac.bi\_t\* hp\_air**

High pressure air OK.

Definition at line 134 of file Ispmac.c.

**7.12.5.32 Ispmac\_motor\_t\* kappa**

Kappa.

Definition at line 115 of file Ispmac.c.

**7.12.5.33 Ispmac.bi\_t\* lp\_air**

Low pressure air OK.

Definition at line 133 of file Ispmac.c.

**7.12.5.34 Ispg\_demandairrights\_t Ispg\_demandairrights**

our demandairrights object

Definition at line 65 of file Ispg.c.

**7.12.5.35 Ispg\_getcenter\_t Ispg\_getcenter**

the getcenter object

Definition at line 64 of file Ispg.c.

**7.12.5.36 Ispg\_getcurrentsampleid\_t Ispg\_getcurrentsampleid**

our currentsample id

Definition at line 66 of file Ispg.c.

**7.12.5.37 `lspg_nexsample_t` `lspg_nexsample`**

the very next sample

Definition at line 62 of file `lspg.c`.

**7.12.5.38 `lspg_nextshot_t` `lspg_nextshot`**

the nextshot object

Definition at line 63 of file `lspg.c`.

**7.12.5.39 `lspg_starttransfer_t` `lspg_starttransfer`**

start a sample transfer

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

**7.12.5.40 `lspg_waitcryo_t` `lspg_waitcryo`**

signal the robot

Definition at line 68 of file `lspg.c`.

**7.12.5.41 `lspmac_motor_t` `lspmac_motors[]`**

All our motors.

Definition at line 98 of file `lspmac.c`.

**7.12.5.42 `pthread_cond_t` `lspmac_moving_cond`**

Wait for motor(s) to finish moving condition.

Definition at line 72 of file `lspmac.c`.

**7.12.5.43 `int` `lspmac_moving_flags`**

Flag used to implement motor moving condition.

Definition at line 73 of file `lspmac.c`.

**7.12.5.44 `pthread_mutex_t` `lspmac_moving_mutex`**

Coordinate moving motors between threads.

Definition at line 71 of file `lspmac.c`.

**7.12.5.45 `int` `lspmac_nmotors`**

The number of motors we manage.

Definition at line 99 of file `lspmac.c`.

**7.12.5.46 pthread\_cond\_t lsppmac\_shutter\_cond**

Allows waiting for the shutter status to change.

Definition at line 70 of file lsppmac.c.

**7.12.5.47 int lsppmac\_shutter\_has\_opened**

Indicates that the shutter had opened, perhaps briefly even if the state did not change.

Definition at line 68 of file lsppmac.c.

**7.12.5.48 pthread\_mutex\_t lsppmac\_shutter\_mutex**

Coordinates threads reading shutter status.

Definition at line 69 of file lsppmac.c.

**7.12.5.49 int lsppmac\_shutter\_state**

State of the shutter, used to detect changes.

Definition at line 67 of file lsppmac.c.

**7.12.5.50 pthread\_cond\_t lsredis\_cond**

Definition at line 75 of file lsredis.c.

**7.12.5.51 pthread\_mutex\_t lsredis\_mutex**

Definition at line 74 of file lsredis.c.

**7.12.5.52 int lsredis\_running**

Definition at line 76 of file lsredis.c.

**7.12.5.53 pthread\_mutex\_t md2\_status\_mutex**

Synchronize reading/writing status buffer.

Definition at line 354 of file lsppmac.c.

**7.12.5.54 char md2cmds\_cmd[]**

our command;

Definition at line 24 of file md2cmds.c.

**7.12.5.55 pthread\_cond\_t md2cmds\_cond**

condition to signal when it's time to run an md2 command

Definition at line 10 of file md2cmds.c.

**7.12.5.56 `lredis_obj_t* md2cmds.md_status_code`**

Definition at line 26 of file md2cmds.c.

**7.12.5.57 `pthread_mutex_t md2cmds_mutex`**

mutex for the condition

Definition at line 11 of file md2cmds.c.

**7.12.5.58 `pthread_cond_t md2cmds_pg_cond`****7.12.5.59 `pthread_mutex_t md2cmds_pg_mutex`****7.12.5.60 `lspmac_bi_t* minikappa_ok`**

Minikappa is OK (whatever that means)

Definition at line 144 of file lspmac.c.

**7.12.5.61 `pthread_mutex_t ncurses_mutex`**

allow more than one thread access to the screen

Definition at line 242 of file pgpmac.c.

**7.12.5.62 `lspmac_motor_t* omega`**

MD2 omega axis (the air bearing)

Definition at line 102 of file lspmac.c.

**7.12.5.63 `struct timespec omega_zero_time`**

Time we believe that omega crossed zero.

Definition at line 82 of file lspmac.c.

**7.12.5.64 `int pgpmac_use_autoscint`**

non-zero to automatically move the alignment stage when the scintillator moves (redis hash AUTOSCINT in config.-HOSTNAME sets this)

Definition at line 247 of file pgpmac.c.

**7.12.5.65 `int pgpmac_use_pg`**

non-zero to start up lsgp thread, 0 to not (reids hash PG in config.HOSTNAME sets this)

Definition at line 246 of file pgpmac.c.

**7.12.5.66 `lspmac_motor_t* phi`**

Phi (not data collection axis)

Definition at line 116 of file lspmac.c.

**7.12.5.67 pthread\_cond\_t pmac\_queue\_cond**

wait for a command to be sent to PMAC before continuing

Definition at line 88 of file lspmacc.c.

**7.12.5.68 pthread\_mutex\_t pmac\_queue\_mutex**

manage access to the pmac command queue

Definition at line 87 of file lspmacc.c.

**7.12.5.69 lspmacc.bi\_t\* sample\_detected**

smart magnet detected sample

Definition at line 140 of file lspmacc.c.

**7.12.5.70 lspmacc.motor\_t\* scint**

Scintillator Z.

Definition at line 112 of file lspmacc.c.

**7.12.5.71 lspmacc.bi\_t\* shutter\_open**

shutter is open (note in pmc says this is a slow input)

Definition at line 147 of file lspmacc.c.

**7.12.5.72 lspmacc.bi\_t\* smart\_mag\_err**

smart magnet error (coil broken perhaps)

Definition at line 148 of file lspmacc.c.

**7.12.5.73 lspmacc.bi\_t\* smart\_mag\_off**

smart magnet is off

Definition at line 149 of file lspmacc.c.

**7.12.5.74 lspmacc.bi\_t\* smart\_mag\_on**

smart magnet is on

Definition at line 145 of file lspmacc.c.

**7.12.5.75 lspmacc.motor\_t\* smart\_mag\_oo**

Smart Magnet on/off.

Definition at line 123 of file lspmacc.c.

**7.12.5.76 WINDOW\* term\_input**

place to put the cursor

Definition at line 238 of file pgpmac.c.

**7.12.5.77 WINDOW\* term\_output**

place to print stuff out

Definition at line 237 of file pgpmac.c.

**7.12.5.78 WINDOW\* term\_status**

shutter, lamp, air, etc status

Definition at line 239 of file pgpmac.c.

**7.12.5.79 WINDOW\* term\_status2**

shutter, lamp, air, etc status

Definition at line 240 of file pgpmac.c.

**7.12.5.80 Ispmac\_motor\_t\* zoom**

Optical zoom.

Definition at line 107 of file Ispmac.c.

# Index

\_GNU\_SOURCE pgpmac.h, 356  
    pgpmac.h, 307  
\_\_init\_\_ iniParser::iniParser, 18  
\_lspmac\_motor\_init lspmac.c, 146  
\_lsredis\_get\_obj lsredis.c, 225  
    pgpmac.h, 309  
\_lsredis\_set\_value lsredis.c, 226  
    acc11c\_1 md2StatusStruct, 66  
    acc11c\_2 md2StatusStruct, 66  
    acc11c\_3 md2StatusStruct, 66  
    acc11c\_5 md2StatusStruct, 66  
    acc11c\_6 md2StatusStruct, 66  
active lspg\_nextshot\_struct, 33  
    lspmac\_motor\_struct, 52  
active2 lspg\_nextshot\_struct, 33  
active2\_isnull lspg\_nextshot\_struct, 33  
active\_init lspmac\_motor\_struct, 52  
active\_isnull lspg\_nextshot\_struct, 33  
active\_simulation mk\_pgpmac\_redis, 12  
actual\_pos\_cnts lspmac\_motor\_struct, 52  
actual\_pos\_cnts\_p lspmac\_motor\_struct, 53  
alignx lspmac.c, 212  
    pgpmac.h, 356  
alignx\_act\_pos md2StatusStruct, 66  
alignx\_status\_1 md2StatusStruct, 66  
alignx\_status\_2 md2StatusStruct, 66  
aligny lspmac.c, 212  
    pgpmac.h, 356  
aligny\_act\_pos md2StatusStruct, 66  
aligny\_status\_1 md2StatusStruct, 66  
aligny\_status\_2 md2StatusStruct, 67  
alignz lspmac.c, 212  
    pgpmac.h, 356  
alignz\_act\_pos md2StatusStruct, 67  
alignz\_status\_1 md2StatusStruct, 67  
alignz\_status\_2 md2StatusStruct, 67  
anal lspmac.c, 212  
    pgpmac.h, 356  
analyzer\_act\_pos md2StatusStruct, 67  
analyzer\_status\_1 md2StatusStruct, 67  
analyzer\_status\_2 md2StatusStruct, 67  
aperturey\_act\_pos md2StatusStruct, 67  
aperturey\_status\_1 md2StatusStruct, 67  
aperturey\_status\_2 md2StatusStruct, 67  
aperturez\_act\_pos md2StatusStruct, 67  
aperturez\_status\_1 md2StatusStruct, 67  
aperturez\_status\_2 md2StatusStruct, 68  
apery lspmac.c, 212  
    pgpmac.h, 356  
aperz lspmac.c, 212  
    pgpmac.h, 356  
arm\_parked lspmac.c, 212  
    pgpmac.h, 356  
asis mk\_pgpmac\_redis, 12  
avalue

lsredis\_obj\_struct, 59  
 ax  
   lspg\_nextshot\_struct, 33  
 ax2  
   lspg\_nextshot\_struct, 33  
 ax2\_isnull  
   lspg\_nextshot\_struct, 33  
 ax\_isnull  
   lspg\_nextshot\_struct, 33  
 axis  
   lspmac\_combined\_move\_struct, 49  
   lspmac\_motor\_struct, 53  
 ay  
   lspg\_nextshot\_struct, 33  
 ay2  
   lspg\_nextshot\_struct, 34  
 ay2\_isnull  
   lspg\_nextshot\_struct, 34  
 ay\_isnull  
   lspg\_nextshot\_struct, 34  
 az  
   lspg\_nextshot\_struct, 34  
 az2  
   lspg\_nextshot\_struct, 34  
 az2\_isnull  
   lspg\_nextshot\_struct, 34  
 az\_isnull  
   lspg\_nextshot\_struct, 34  
  
 b  
   mk\_pgpmac\_redis, 13  
 bData  
   tagEthernetCmd, 72  
 back\_dac  
   md2StatusStruct, 68  
 bi\_list  
   mk\_pgpmac\_redis, 13  
 blight  
   lspmac.c, 212  
   pgpmac.h, 356  
 blight\_down  
   lspmac.c, 212  
   pgpmac.h, 356  
 blight\_f  
   lspmac.c, 213  
   pgpmac.h, 356  
 blight\_ud  
   lspmac.c, 213  
   pgpmac.h, 357  
 blight\_up  
   lspmac.c, 213  
   pgpmac.h, 357  
 bvalue  
   lsredis\_obj\_struct, 59  
  
 capy  
   lspmac.c, 213  
   pgpmac.h, 357  
 capy\_act\_pos  
     md2StatusStruct, 68  
     capy\_status\_1  
       md2StatusStruct, 68  
     capy\_status\_2  
       md2StatusStruct, 68  
 capz  
   lspmac.c, 213  
   pgpmac.h, 357  
 capz\_act\_pos  
     md2StatusStruct, 68  
     capy\_status\_1  
       md2StatusStruct, 68  
     capy\_status\_2  
       md2StatusStruct, 68  
 cb  
   lsevents\_callbacks\_struct, 20  
   lsevents\_listener\_struct, 21  
 cbl  
   lsevents\_event\_names\_struct, 20  
 centerx\_act\_pos  
     md2StatusStruct, 68  
 centerx\_status\_1  
     md2StatusStruct, 68  
 centerx\_status\_2  
     md2StatusStruct, 68  
 centery\_act\_pos  
     md2StatusStruct, 68  
 centery\_status\_1  
     md2StatusStruct, 69  
 centery\_status\_2  
     md2StatusStruct, 69  
 cenx  
   lspmac.c, 213  
   pgpmac.h, 357  
 ceny  
   lspmac.c, 213  
   pgpmac.h, 357  
 changeEventOff  
   lspmac\_bi\_struct, 47  
 changeEventOn  
   lspmac\_bi\_struct, 47  
 cleanstr  
   lspmac.c, 147  
 command\_buf  
   lspmac\_ascii\_buffers\_struct, 45  
 command\_buf\_cc  
   lspmac\_ascii\_buffers\_struct, 45  
 command\_sent  
   lspmac\_motor\_struct, 53  
 command\_str  
   lspmac\_ascii\_buffers\_struct, 45  
 cond  
   lspg\_demandairrights\_struct, 23  
   lspg\_getcenter\_struct, 24  
   lspg\_getcurrentsampleid\_struct, 27  
   lspg\_lock\_detector\_struct, 28  
   lspg\_lock\_diffractometer\_struct, 28  
   lspg\_nexsample\_struct, 29

lspg\_nextshot\_struct, 34  
lspg\_seq\_run\_prep\_struct, 41  
lspg\_starttransfer\_struct, 42  
lspg\_wait\_for\_detector\_struct, 43  
lspg\_waitcryo\_struct, 44  
lspmac\_motor\_struct, 53  
lsredis\_obj\_struct, 59  
configs  
  mk\_pgpmac\_redis, 13  
coord\_num  
  lspmac\_combined\_move\_struct, 49  
  lspmac\_motor\_struct, 53  
cr\_cmd  
  lspmac.c, 213  
cryo  
  lspmac.c, 213  
  pgpmac.h, 357  
cryo\_back  
  lspmac.c, 214  
  pgpmac.h, 357  
cryo\_switch  
  lspmac.c, 214  
  pgpmac.h, 357  
cvalue  
  lsredis\_obj\_struct, 59  
cx  
  lspg\_nextshot\_struct, 34  
cx2  
  lspg\_nextshot\_struct, 34  
cx2\_isnull  
  lspg\_nextshot\_struct, 35  
cx\_isnull  
  lspg\_nextshot\_struct, 35  
cy  
  lspg\_nextshot\_struct, 35  
cy2  
  lspg\_nextshot\_struct, 35  
cy2\_isnull  
  lspg\_nextshot\_struct, 35  
cy\_isnull  
  lspg\_nextshot\_struct, 35  
dac\_mvar  
  lspmac\_motor\_struct, 53  
dax  
  lspg\_getcenter\_struct, 24  
dax\_isnull  
  lspg\_getcenter\_struct, 25  
day  
  lspg\_getcenter\_struct, 25  
day\_isnull  
  lspg\_getcenter\_struct, 25  
daz  
  lspg\_getcenter\_struct, 25  
daz\_isnull  
  lspg\_getcenter\_struct, 25  
dbmem  
  lspmac.c, 214  
dbmemln

  lspmac.c, 214  
dcx  
  lspg\_getcenter\_struct, 25  
dcx\_isnull  
  lspg\_getcenter\_struct, 25  
dcy  
  lspg\_getcenter\_struct, 25  
dcy\_isnull  
  lspg\_getcenter\_struct, 25  
delay\_nsecs  
  ltimer\_list\_struct, 62  
delay\_secs  
  ltimer\_list\_struct, 62  
Delta  
  lspmac\_combined\_move\_struct, 49  
doomsday\_count  
  pgpmac.c, 297  
doomsday\_mutex  
  pgpmac.c, 297  
dryer  
  lspmac.c, 214  
  pgpmac.h, 358  
dsdir  
  lspg\_nextshot\_struct, 35  
dsdir\_isnull  
  lspg\_nextshot\_struct, 35  
dsdist  
  lspg\_nextshot\_struct, 35  
dsdist2  
  lspg\_nextshot\_struct, 35  
dsdist2\_isnull  
  lspg\_nextshot\_struct, 36  
dsdist\_isnull  
  lspg\_nextshot\_struct, 36  
dsexp  
  lspg\_nextshot\_struct, 36  
dsexp2  
  lspg\_nextshot\_struct, 36  
dsexp2\_isnull  
  lspg\_nextshot\_struct, 36  
dsexp\_isnull  
  lspg\_nextshot\_struct, 36  
dshpid  
  lspg\_nextshot\_struct, 36  
dshpid\_isnull  
  lspg\_nextshot\_struct, 36  
dskappa  
  lspg\_nextshot\_struct, 36  
dskappa2  
  lspg\_nextshot\_struct, 36  
dskappa2\_isnull  
  lspg\_nextshot\_struct, 37  
dskappa\_isnull  
  lspg\_nextshot\_struct, 37  
dsnrg  
  lspg\_nextshot\_struct, 37  
dsnrg2  
  lspg\_nextshot\_struct, 37

dsnrg2\_isnull  
     lspg\_nextshot\_struct, 37  
 dsnrg\_isnull  
     lspg\_nextshot\_struct, 37  
 dsomega  
     lspg\_nextshot\_struct, 37  
 dsomega2  
     lspg\_nextshot\_struct, 37  
 dsomega2\_isnull  
     lspg\_nextshot\_struct, 37  
 dsomega\_isnull  
     lspg\_nextshot\_struct, 37  
 dsoscaxis  
     lspg\_nextshot\_struct, 37  
 dsoscaxis2  
     lspg\_nextshot\_struct, 38  
 dsoscaxis2\_isnull  
     lspg\_nextshot\_struct, 38  
 dsoscaxis\_isnull  
     lspg\_nextshot\_struct, 38  
 dsowidth  
     lspg\_nextshot\_struct, 38  
 dsowidth2  
     lspg\_nextshot\_struct, 38  
 dsowidth2\_isnull  
     lspg\_nextshot\_struct, 38  
 dsowidth\_isnull  
     lspg\_nextshot\_struct, 38  
 dphi  
     lspg\_nextshot\_struct, 38  
 dphi2  
     lspg\_nextshot\_struct, 38  
 dphi2\_isnull  
     lspg\_nextshot\_struct, 38  
 dphi\_isnull  
     lspg\_nextshot\_struct, 39  
 dpid  
     lspg\_nextshot\_struct, 39  
 dpid\_isnull  
     lspg\_nextshot\_struct, 39  
 dummy1  
     md2StatusStruct, 69  
 dummy2  
     md2StatusStruct, 69  
 dummy3  
     md2StatusStruct, 69  
 dummy4  
     md2StatusStruct, 69  
 dummy5  
     md2StatusStruct, 69  
 dummy6  
     md2StatusStruct, 69  
 dummy7  
     md2StatusStruct, 69  
 dummy8  
     md2StatusStruct, 69  
 dummy9  
     md2StatusStruct, 69

dummyA  
     md2StatusStruct, 69  
 dummyB  
     md2StatusStruct, 70  
 dvalue  
     lsredis\_obj\_struct, 59

etel\_init\_ok  
     lspmac.c, 214  
     pgpmac.h, 358  
 etel\_on  
     lspmac.c, 214  
     pgpmac.h, 358  
 etel\_ready  
     lspmac.c, 214  
     pgpmac.h, 358  
 ethCmdOff  
     lspmac.c, 214  
 ethCmdOn  
     lspmac.c, 215  
 ethCmdQueue  
     lspmac.c, 215  
 ethCmdReply  
     lspmac.c, 215  
 event  
     lsevents\_event\_names\_struct, 20  
     lspmac\_cmd\_queue\_struct, 48  
     lspmac\_dpascii\_queue\_struct, 50  
     lstimer\_list\_struct, 63  
 events\_name  
     lsredis\_obj\_struct, 60

evp  
     lsevents\_queue\_struct, 22

f  
     iniParser::iniParser, 19  
     mk\_pgpmac\_redis, 14

first\_time  
     lspmac\_bi\_struct, 47

flight  
     lspmac.c, 215  
     pgpmac.h, 358  
 flight\_f  
     lspmac.c, 215  
     pgpmac.h, 358  
 flight\_oo  
     lspmac.c, 215  
     pgpmac.h, 358  
 fluo  
     lspmac.c, 215  
     pgpmac.h, 358  
 fluor\_back  
     lspmac.c, 215  
     pgpmac.h, 358  
 fnc  
     mk\_pgpmac\_redis, 14

front\_dac  
     md2StatusStruct, 70

fs\_has\_opened

md2StatusStruct, 70  
fs\_has\_opened\_globally  
    md2StatusStruct, 70  
fs\_is\_open  
    md2StatusStruct, 70  
fscint  
    lspmac.c, 215  
    pgpmac.h, 359  
fshut  
    lspmac.c, 216  
    pgpmac.h, 359  
  
gb\_cmd  
    lspmac.c, 216  
get  
    iniParser::iniParser, 18  
getcurrentsampleid  
    lspg\_getcurrentsampleid\_struct, 27  
getcurrentsampleid\_isnull  
    lspg\_getcurrentsampleid\_struct, 27  
getivars  
    lspmac.c, 216  
getmvars  
    lspmac.c, 216  
  
handler  
    lstimer.c, 250  
hard\_ini  
    mk\_pgpmac\_redis, 14  
hard\_ini\_fields  
    mk\_pgpmac\_redis, 14  
has\_option  
    iniParser::iniParser, 18  
has\_section  
    iniParser::iniParser, 18  
head  
    mk\_pgpmac\_redis, 14  
hex\_dump  
    lspmac.c, 147  
hi  
    mk\_pgpmac\_redis, 14  
hits  
    lsredis\_obj\_struct, 60  
home  
    lspmac\_motor\_struct, 53  
homing  
    lspmac\_motor\_struct, 53  
hp\_air  
    lspmac.c, 216  
    pgpmac.h, 359  
  
i  
    mk\_pgpmac\_redis, 14  
in\_position\_band  
    lspmac\_motor\_struct, 53  
inactive\_init  
    lspmac\_motor\_struct, 54  
index  
    lsredis\_preset\_list\_struct, 61  
  
    iniParser, 11  
        ip, 11  
    iniParser.iniParser, 17  
    iniParser.py, 75  
    iniParser::iniParser  
        \_\_init\_\_, 18  
        f, 19  
        get, 18  
        has\_option, 18  
        has\_section, 18  
        options, 18  
        read, 18  
        sd, 19  
        sections, 19  
    init\_nsecs  
        lstimer\_list\_struct, 63  
    init\_secs  
        lstimer\_list\_struct, 63  
    ip  
        iniParser, 11  
  
jogAbs  
    lspmac\_motor\_struct, 54  
  
k  
    md2cmds\_cmd\_kv\_struct, 64  
kappa  
    lspmac.c, 216  
    pgpmac.h, 359  
kappa\_act\_pos  
    md2StatusStruct, 70  
kappa\_status\_1  
    md2StatusStruct, 70  
kappa\_status\_2  
    md2StatusStruct, 70  
key  
    lsredis\_obj\_struct, 60  
    lsredis\_preset\_list\_struct, 61  
  
    LS\_PG\_STATE\_IDLE  
        lspg.c, 94  
    LS\_PG\_STATE\_INIT  
        lspg.c, 94  
    LS\_PG\_STATE\_RECV  
        lspg.c, 94  
    LS\_PG\_STATE\_RESET  
        lspg.c, 95  
    LS\_PG\_STATE\_SEND  
        lspg.c, 95  
    LS\_PMAC\_STATE\_CR  
        lspmac.c, 142  
    LS\_PMAC\_STATE\_GB  
        lspmac.c, 142  
    LS\_PMAC\_STATE\_GMR  
        lspmac.c, 142  
    LS\_PMAC\_STATE\_IDLE  
        lspmac.c, 142  
    LS\_PMAC\_STATE\_RESET  
        lspmac.c, 143

LS\_PMAC\_STATE\_RR  
     lspmac.c, 143  
 LS\_PMAC\_STATE\_SC  
     lspmac.c, 143  
 LS\_PMAC\_STATE\_WACK  
     lspmac.c, 143  
 LS\_PMAC\_STATE\_WCR  
     lspmac.c, 143  
 LS\_PMAC\_STATE\_WGB  
     lspmac.c, 143  
 LSLOGGING\_FILE\_NAME  
     lslogging.c, 85  
 LSPMAC\_MAGIC\_NUMBER  
     pgpmac.h, 307  
 LSPMAC\_MAX\_MOTORS  
     lspmac.c, 143  
 LSPMAC\_PRESET\_REGEX  
     lspmac.c, 143  
 LSTIMER\_LIST\_LENGTH  
     lstimer.c, 249  
 last\_nsecs  
     lstimer\_list\_struct, 63  
 last\_secs  
     lstimer\_list\_struct, 63  
 lmsg  
     lslogging\_queue\_struct, 23  
 lp\_air  
     lspmac.c, 216  
     pgpmac.h, 359  
 ls\_pg\_state  
     lspg.c, 130  
 ls\_pmac\_state  
     lspmac.c, 216  
 lsConnect  
     lspmac.c, 148  
 lsevents.c, 75  
     lsevents\_add\_listener, 78  
     lsevents\_callbacks\_t, 77  
     lsevents\_event\_name\_ht, 83  
     lsevents\_event\_names, 83  
     lsevents\_event\_names\_t, 77  
     lsevents\_init, 78  
     lsevents\_listener\_mutex, 83  
     lsevents\_listener\_t, 77  
     lsevents\_listeners\_p, 83  
     lsevents\_max\_events, 83  
     lsevents\_n\_events, 83  
     lsevents\_preregister\_event, 79  
     lsevents\_queue, 83  
     lsevents\_queue\_cond, 83  
     lsevents\_queue\_mutex, 83  
     lsevents\_queue\_off, 83  
     lsevents\_queue\_on, 84  
     lsevents\_queue\_t, 77  
     lsevents\_register\_event, 79  
     lsevents\_remove\_listener, 80  
     lsevents\_run, 81  
     lsevents\_send\_event, 81  
     lsevents\_thread, 84  
     lsevents\_worker, 82  
 lsevents\_add\_listener  
     lsevents.c, 78  
     pgpmac.h, 310  
 lsevents\_callbacks\_struct, 19  
     cb, 20  
     next, 20  
 lsevents\_callbacks\_t  
     lsevents.c, 77  
 lsevents\_event\_name\_ht  
     lsevents.c, 83  
 lsevents\_event\_names  
     lsevents.c, 83  
 lsevents\_event\_names\_struct, 20  
     tbl, 20  
     event, 20  
     next, 20  
 lsevents\_event\_names\_t  
     lsevents.c, 77  
 lsevents\_init  
     lsevents.c, 78  
     pgpmac.h, 311  
 lsevents\_listener\_mutex  
     lsevents.c, 83  
 lsevents\_listener\_struct, 21  
     cb, 21  
     next, 21  
     raw\_regex, 21  
     re, 21  
 lsevents\_listener\_t  
     lsevents.c, 77  
 lsevents\_listeners\_p  
     lsevents.c, 83  
 lsevents\_max\_events  
     lsevents.c, 83  
 lsevents\_n\_events  
     lsevents.c, 83  
 lsevents\_preregister\_event  
     lsevents.c, 79  
     pgpmac.h, 311  
 lsevents\_queue  
     lsevents.c, 83  
 lsevents\_queue\_cond  
     lsevents.c, 83  
 lsevents\_queue\_mutex  
     lsevents.c, 83  
 lsevents\_queue\_off  
     lsevents.c, 83  
 lsevents\_queue\_on  
     lsevents.c, 84  
 lsevents\_queue\_struct, 22  
     evp, 22  
 lsevents\_queue\_t  
     lsevents.c, 77  
 lsevents\_register\_event  
     lsevents.c, 79  
 lsevents\_remove\_listener

lsevents.c, 80  
pgpmac.h, 311  
lsevents\_run  
    lsevents.c, 81  
    pgpmac.h, 312  
lsevents\_send\_event  
    lsevents.c, 81  
    pgpmac.h, 313  
lsevents\_thread  
    lsevents.c, 84  
lsevents\_worker  
    lsevents.c, 82  
lslogging.c, 84  
    LSLOGGING\_FILE\_NAME, 85  
    lslogging\_cond, 88  
    lslogging\_event\_cb, 86  
    lslogging\_file, 88  
    lslogging\_init, 86  
    lslogging\_log\_message, 86  
    lslogging\_mutex, 88  
    lslogging\_off, 88  
    lslogging\_on, 88  
    lslogging\_queue, 88  
    lslogging\_queue\_t, 86  
    lslogging\_run, 87  
    lslogging\_thread, 88  
    lslogging\_worker, 87  
lslogging\_cond  
    lslogging.c, 88  
lslogging\_event\_cb  
    lslogging.c, 86  
lslogging\_file  
    lslogging.c, 88  
lslogging\_init  
    lslogging.c, 86  
    pgpmac.h, 313  
lslogging\_log\_message  
    lslogging.c, 86  
    pgpmac.h, 313  
lslogging\_mutex  
    lslogging.c, 88  
lslogging\_off  
    lslogging.c, 88  
lslogging\_on  
    lslogging.c, 88  
lslogging\_queue  
    lslogging.c, 88  
lslogging\_queue\_struct, 22  
    lmsg, 23  
    ltime, 23  
lslogging\_queue\_t  
    lslogging.c, 86  
lslogging\_run  
    lslogging.c, 87  
    pgpmac.h, 314  
lslogging\_thread  
    lslogging.c, 88  
lslogging\_worker  
    lslogging.c, 84  
lslogging.c, 87  
lsevents.c, 89  
    LS\_PG\_STATE\_IDLE, 94  
    LS\_PG\_STATE\_INIT, 94  
    LS\_PG\_STATE\_RECV, 94  
    LS\_PG\_STATE\_RESET, 95  
    LS\_PG\_STATE\_SEND, 95  
    ls\_pg\_state, 130  
    lspg\_allkvs\_cb, 95  
    lspg\_array2ptrs, 96  
    lspg\_check\_preset\_in\_position\_cb, 97  
    lspg\_cmd\_cb, 98  
    lspg\_connectPoll\_response, 130  
    lspg\_demandairrights, 130  
    lspg\_demandairrights\_all, 98  
    lspg\_demandairrights\_call, 98  
    lspg\_demandairrights\_cb, 99  
    lspg\_demandairrights\_init, 99  
    lspg\_demandairrights\_wait, 99  
    lspg\_flush, 99  
    lspg\_getcenter, 130  
    lspg\_getcenter\_all, 100  
    lspg\_getcenter\_call, 100  
    lspg\_getcenter\_cb, 100  
    lspg\_getcenter\_done, 101  
    lspg\_getcenter\_init, 101  
    lspg\_getcenter\_wait, 101  
    lspg\_getcurrentsampleid, 130  
    lspg\_getcurrentsampleid\_call, 102  
    lspg\_getcurrentsampleid\_cb, 102  
    lspg\_getcurrentsampleid\_init, 102  
    lspg\_getcurrentsampleid\_read, 103  
    lspg\_getcurrentsampleid\_wait\_for\_id, 103  
    lspg\_init, 103  
    lspg\_lock\_detector, 130  
    lspg\_lock\_detector\_all, 103  
    lspg\_lock\_detector\_call, 104  
    lspg\_lock\_detector\_cb, 104  
    lspg\_lock\_detector\_done, 104  
    lspg\_lock\_detector\_init, 104  
    lspg\_lock\_detector\_t, 95  
    lspg\_lock\_detector\_wait, 104  
    lspg\_lock\_diffractometer, 130  
    lspg\_lock\_diffractometer\_all, 105  
    lspg\_lock\_diffractometer\_call, 105  
    lspg\_lock\_diffractometer\_cb, 105  
    lspg\_lock\_diffractometer\_done, 105  
    lspg\_lock\_diffractometer\_init, 106  
    lspg\_lock\_diffractometer\_t, 95  
    lspg\_lock\_diffractometer\_wait, 106  
    lspg\_next\_state, 106  
    lspg\_nexaction\_cb, 107  
    lspg\_nexerrors\_cb, 107  
    lspg\_nexsample, 131  
    lspg\_nexsample\_all, 108  
    lspg\_nexsample\_call, 108  
    lspg\_nexsample\_cb, 108  
    lspg\_nexsample\_done, 109

lsgp\_nextsample\_init, 109  
 lsgp\_nextsample\_wait, 109  
 lsgp\_nextshot, 131  
 lsgp\_nextshot\_call, 110  
 lsgp\_nextshot\_cb, 110  
 lsgp\_nextshot\_done, 114  
 lsgp\_nextshot\_init, 114  
 lsgp\_nextshot\_wait, 114  
 lsgp\_notice\_processor, 114  
 lsgp\_pg\_connect, 114  
 lsgp\_pg\_service, 116  
 lsgp\_preset\_changed\_cb, 117  
 lsgp\_query\_next, 118  
 lsgp\_query\_push, 118  
 lsgp\_query\_queue, 131  
 lsgp\_query\_queue\_off, 131  
 lsgp\_query\_queue\_on, 131  
 lsgp\_query\_queue\_reply, 131  
 lsgp\_query\_reply\_next, 118  
 lsgp\_query\_reply\_peek, 119  
 lsgp\_queue\_cond, 131  
 lsgp\_queue\_mutex, 131  
 lsgp\_quitting\_cb, 119  
 lsgp\_receive, 119  
 lsgp\_resetPoll\_response, 131  
 lsgp\_run, 120  
 lsgp\_sample\_detector\_cb, 121  
 lsgp\_send\_next\_query, 121  
 lsgp\_seq\_run\_prep, 132  
 lsgp\_seq\_run\_prep\_all, 121  
 lsgp\_seq\_run\_prep\_call, 122  
 lsgp\_seq\_run\_prep\_cb, 122  
 lsgp\_seq\_run\_prep\_done, 123  
 lsgp\_seq\_run\_prep\_init, 123  
 lsgp\_seq\_run\_prep\_t, 95  
 lsgp\_seq\_run\_prep\_wait, 123  
 lsgp\_set\_scale\_cb, 123  
 lsgp\_sig\_service, 124  
 lsgp\_starttransfer, 132  
 lsgp\_starttransfer\_all, 124  
 lsgp\_starttransfer\_call, 124  
 lsgp\_starttransfer\_cb, 125  
 lsgp\_starttransfer\_done, 125  
 lsgp\_starttransfer\_init, 125  
 lsgp\_starttransfer\_wait, 125  
 lsgp\_thread, 132  
 lsgp\_unset\_current\_preset\_moving\_cb, 126  
 lsgp\_update\_kvs\_cb, 126  
 lsgp\_wait\_for\_detector, 132  
 lsgp\_wait\_for\_detector\_all, 127  
 lsgp\_wait\_for\_detector\_call, 127  
 lsgp\_wait\_for\_detector\_cb, 127  
 lsgp\_wait\_for\_detector\_done, 127  
 lsgp\_wait\_for\_detector\_init, 128  
 lsgp\_wait\_for\_detector\_t, 95  
 lsgp\_wait\_for\_detector\_wait, 128  
 lsgp\_waitcryo, 132  
 lsgp\_waitcryo\_all, 128  
 lsgp\_waitcryo\_cb, 128  
 lsgp\_waitcryo\_init, 129  
 lsgp\_worker, 129  
 lsgfd, 132  
 now, 132  
 q, 132  
 lsgp\_allkvs\_cb  
 lsgp.c, 95  
 lsgp\_array2ptrs  
 lsgp.c, 96  
 pgpmac.h, 314  
 lsgp\_check\_preset\_in\_position\_cb  
 lsgp.c, 97  
 lsgp\_cmd\_cb  
 lsgp.c, 98  
 lsgp\_connectPoll\_response  
 lsgp.c, 130  
 lsgp\_demandairrights  
 lsgp.c, 130  
 pgpmac.h, 359  
 lsgp\_demandairrights\_all  
 lsgp.c, 98  
 pgpmac.h, 316  
 lsgp\_demandairrights\_call  
 lsgp.c, 98  
 lsgp\_demandairrights\_cb  
 lsgp.c, 99  
 lsgp\_demandairrights\_init  
 lsgp.c, 99  
 lsgp\_demandairrights\_struct, 23  
 cond, 23  
 mutex, 23  
 new\_value\_ready, 23  
 lsgp\_demandairrights\_t  
 pgpmac.h, 308  
 lsgp\_demandairrights\_wait  
 lsgp.c, 99  
 lsgp\_flush  
 lsgp.c, 99  
 lsgp\_getcenter  
 lsgp.c, 130  
 pgpmac.h, 359  
 lsgp\_getcenter\_all  
 lsgp.c, 100  
 lsgp\_getcenter\_call  
 lsgp.c, 100  
 pgpmac.h, 316  
 lsgp\_getcenter\_cb  
 lsgp.c, 100  
 lsgp\_getcenter\_done  
 lsgp.c, 101  
 pgpmac.h, 316  
 lsgp\_getcenter\_init  
 lsgp.c, 101  
 lsgp\_getcenter\_struct, 24  
 cond, 24  
 dax, 24  
 dax\_isnull, 25

day, 25  
day\_isnull, 25  
daz, 25  
daz\_isnull, 25  
dcx, 25  
dcx\_isnull, 25  
dcy, 25  
dcy\_isnull, 25  
mutex, 25  
new\_value\_ready, 26  
no\_rows\_returned, 26  
zoom, 26  
zoom\_isnull, 26  
lspg\_getcenter\_t  
  pgpmac.h, 308  
lspg\_getcenter\_wait  
  lspg.c, 101  
  pgpmac.h, 316  
lspg\_getcurrentsampleid  
  lspg.c, 130  
  pgpmac.h, 359  
lspg\_getcurrentsampleid\_call  
  lspg.c, 102  
lspg\_getcurrentsampleid\_cb  
  lspg.c, 102  
lspg\_getcurrentsampleid\_init  
  lspg.c, 102  
lspg\_getcurrentsampleid\_read  
  lspg.c, 103  
lspg\_getcurrentsampleid\_struct, 26  
  cond, 27  
  getcurrentsampleid, 27  
  getcurrentsampleid\_isnull, 27  
  mutex, 27  
  new\_value\_ready, 27  
  no\_rows\_returned, 27  
lspg\_getcurrentsampleid\_t  
  pgpmac.h, 308  
lspg\_getcurrentsampleid\_wait\_for\_id  
  lspg.c, 103  
  pgpmac.h, 316  
lspg\_init  
  lspg.c, 103  
  pgpmac.h, 317  
lspg\_lock\_detector  
  lspg.c, 130  
lspg\_lock\_detector\_all  
  lspg.c, 103  
lspg\_lock\_detector\_call  
  lspg.c, 104  
lspg\_lock\_detector\_cb  
  lspg.c, 104  
lspg\_lock\_detector\_done  
  lspg.c, 104  
lspg\_lock\_detector\_init  
  lspg.c, 104  
lspg\_lock\_detector\_struct, 27  
  cond, 28  
  mutex, 28  
  new\_value\_ready, 28  
  lspg.lock\_detector\_t  
    lspg.c, 95  
  lspg.lock\_detector\_wait  
    lspg.c, 104  
  lspg.lock\_diffractometer  
    lspg.c, 130  
  lspg.lock\_diffractometer\_all  
    lspg.c, 105  
  lspg.lock\_diffractometer\_call  
    lspg.c, 105  
  lspg.lock\_diffractometer\_cb  
    lspg.c, 105  
  lspg.lock\_diffractometer\_done  
    lspg.c, 105  
  lspg.lock\_diffractometer\_init  
    lspg.c, 106  
  lspg.lock\_diffractometer\_struct, 28  
    cond, 28  
    mutex, 28  
    new\_value\_ready, 28  
  lspg.lock\_diffractometer\_t  
    lspg.c, 95  
  lspg.lock\_diffractometer\_wait  
    lspg.c, 106  
lspg.next\_state  
  lspg.c, 106  
lspg.nextaction\_cb  
  lspg.c, 107  
lspg.nexterrors\_cb  
  lspg.c, 107  
lspg.nextsample  
  lspg.c, 131  
  pgpmac.h, 359  
lspg.nextsample\_all  
  lspg.c, 108  
  pgpmac.h, 317  
lspg.nextsample\_call  
  lspg.c, 108  
lspg.nextsample\_cb  
  lspg.c, 108  
lspg.nextsample\_done  
  lspg.c, 109  
lspg.nextsample\_init  
  lspg.c, 109  
lspg.nextsample\_struct, 29  
  cond, 29  
  mutex, 29  
  new\_value\_ready, 29  
  nextsample, 29  
  nextsample\_isnull, 30  
  no\_rows\_returned, 30  
lspg.nextsample\_t  
  pgpmac.h, 308  
lspg.nextsample\_wait  
  lspg.c, 109  
lspg.nextshot

lsgc.c, 131  
 pgpmac.h, 360  
 lsgc\_nextshot\_call  
     lsgc.c, 110  
     pgpmac.h, 317  
 lsgc\_nextshot\_cb  
     lsgc.c, 110  
 lsgc\_nextshot\_done  
     lsgc.c, 114  
     pgpmac.h, 318  
 lsgc\_nextshot\_init  
     lsgc.c, 114  
 lsgc\_nextshot\_struct, 30  
     active, 33  
     active2, 33  
     active2\_isnull, 33  
     active\_isnull, 33  
     ax, 33  
     ax2, 33  
     ax2\_isnull, 33  
     ax\_isnull, 33  
     ay, 33  
     ay2, 34  
     ay2\_isnull, 34  
     ay\_isnull, 34  
     az, 34  
     az2, 34  
     az2\_isnull, 34  
     az\_isnull, 34  
     cond, 34  
     cx, 34  
     cx2, 34  
     cx2\_isnull, 35  
     cx\_isnull, 35  
     cy, 35  
     cy2, 35  
     cy2\_isnull, 35  
     cy\_isnull, 35  
     dsdir, 35  
     dsdir\_isnull, 35  
     dsdist, 35  
     dsdist2, 35  
     dsdist2\_isnull, 36  
     dsdist\_isnull, 36  
     dsexp, 36  
     dsexp2, 36  
     dsexp2\_isnull, 36  
     dsexp\_isnull, 36  
     dsid, 36  
     dsid\_isnull, 36  
     dskappa, 36  
     dskappa2, 36  
     dskappa2\_isnull, 37  
     dskappa\_isnull, 37  
     dsnrg, 37  
     dsnrg2, 37  
     dsnrg2\_isnull, 37  
     dsnrg\_isnull, 37  
     dsomega, 37  
     dsomega2, 37  
     dsomega2\_isnull, 37  
     dsomega\_isnull, 37  
     dsoscaxis, 37  
     dsoscaxis2, 38  
     dsoscaxis2\_isnull, 38  
     dsoscaxis\_isnull, 38  
     dsowidth, 38  
     dsowidth2, 38  
     dsowidth2\_isnull, 38  
     dsowidth\_isnull, 38  
     dsphi, 38  
     dsphi2, 38  
     dsphi2\_isnull, 38  
     dsphi\_isnull, 39  
     dspid, 39  
     dspid\_isnull, 39  
     mutex, 39  
     new\_value\_ready, 39  
     no\_rows\_returned, 39  
     sfm, 39  
     sfm\_isnull, 39  
     sindex, 39  
     sindex2, 39  
     sindex2\_isnull, 40  
     sindex\_isnull, 40  
     skey, 40  
     skey\_isnull, 40  
     sstart, 40  
     sstart2, 40  
     sstart2\_isnull, 40  
     sstart\_isnull, 40  
     stype, 40  
     stype2, 40  
     stype2\_isnull, 41  
     stype\_isnull, 41  
 lsgc\_nextshot\_t  
     pgpmac.h, 308  
 lsgc\_nextshot\_wait  
     lsgc.c, 114  
     pgpmac.h, 318  
 lsgc\_notice\_processor  
     lsgc.c, 114  
 lsgc\_pg\_connect  
     lsgc.c, 114  
 lsgc\_pg\_service  
     lsgc.c, 116  
 lsgc\_preset\_changed\_cb  
     lsgc.c, 117  
 lsgc\_query\_next  
     lsgc.c, 118  
 lsgc\_query\_push  
     lsgc.c, 118  
     pgpmac.h, 318  
 lsgc\_query\_queue  
     lsgc.c, 131  
 lsgc\_query\_queue\_off

lspg.c, 131  
lspg\_query\_queue\_on  
    lspg.c, 131  
lspg\_query\_queue\_reply  
    lspg.c, 131  
lspg\_query\_queue\_t  
    pgpmac.h, 308  
lspg\_query\_reply\_next  
    lspg.c, 118  
lspg\_query\_reply\_peek  
    lspg.c, 119  
lspg\_queue\_cond  
    lspg.c, 131  
lspg\_queue\_mutex  
    lspg.c, 131  
lspg\_quitting\_cb  
    lspg.c, 119  
lspg\_receive  
    lspg.c, 119  
lspg\_resetPoll\_response  
    lspg.c, 131  
lspg\_run  
    lspg.c, 120  
    pgpmac.h, 319  
lspg\_sample\_detector\_cb  
    lspg.c, 121  
lspg\_send\_next\_query  
    lspg.c, 121  
lspg\_seq\_run\_prep  
    lspg.c, 132  
lspg\_seq\_run\_prep\_all  
    lspg.c, 121  
    pgpmac.h, 319  
lspg\_seq\_run\_prep\_call  
    lspg.c, 122  
lspg\_seq\_run\_prep\_cb  
    lspg.c, 122  
lspg\_seq\_run\_prep\_done  
    lspg.c, 123  
lspg\_seq\_run\_prep\_init  
    lspg.c, 123  
lspg\_seq\_run\_prep\_struct, 41  
    cond, 41  
    mutex, 41  
    new\_value\_ready, 41  
lspg\_seq\_run\_prep\_t  
    lspg.c, 95  
lspg\_seq\_run\_prep\_wait  
    lspg.c, 123  
lspg\_set\_scale\_cb  
    lspg.c, 123  
lspg\_sig\_service  
    lspg.c, 124  
lspg\_starttransfer  
    lspg.c, 132  
    pgpmac.h, 360  
lspg\_starttransfer\_all  
    lspg.c, 124  
lspg\_starttransfer\_call  
    lspg.c, 124  
    pgpmac.h, 319  
lspg\_starttransfer\_cb  
    lspg.c, 125  
lspg\_starttransfer\_done  
    lspg.c, 125  
    pgpmac.h, 320  
lspg\_starttransfer\_init  
    lspg.c, 125  
lspg\_starttransfer\_struct, 42  
    cond, 42  
    mutex, 42  
    new\_value\_ready, 42  
    no\_rows\_returned, 42  
    starttransfer, 42  
lspg\_starttransfer\_t  
    pgpmac.h, 308  
lspg\_starttransfer\_wait  
    lspg.c, 125  
    pgpmac.h, 320  
lspg\_thread  
    lspg.c, 132  
lspg\_unset\_current\_preset\_moving\_cb  
    lspg.c, 126  
lspg\_update\_kvs\_cb  
    lspg.c, 126  
lspg\_wait\_for\_detector  
    lspg.c, 132  
lspg\_wait\_for\_detector\_all  
    lspg.c, 127  
lspg\_wait\_for\_detector\_call  
    lspg.c, 127  
lspg\_wait\_for\_detector\_cb  
    lspg.c, 127  
lspg\_wait\_for\_detector\_done  
    lspg.c, 127  
lspg\_wait\_for\_detector\_init  
    lspg.c, 128  
lspg\_wait\_for\_detector\_struct, 43  
    cond, 43  
    mutex, 43  
    new\_value\_ready, 43  
lspg\_wait\_for\_detector\_t  
    lspg.c, 95  
lspg\_wait\_for\_detector\_wait  
    lspg.c, 128  
lspg\_waitcryo  
    lspg.c, 132  
    pgpmac.h, 360  
lspg\_waitcryo\_all  
    lspg.c, 128  
    pgpmac.h, 320  
lspg\_waitcryo\_cb  
    lspg.c, 128  
    pgpmac.h, 320  
lspg\_waitcryo\_init  
    lspg.c, 129

lsgc\_waitcryo\_struct, 43  
 cond, 44  
 mutex, 44  
 new\_value\_ready, 44  
 lsgc\_waitcryo\_t  
 pgpmac.h, 308  
 lsgc\_worker  
 lsgc.c, 129  
 lsgc\_zoom\_lut\_call  
 pgpmac.h, 320  
 lsgcQueryQueueStruct, 44  
 onResponse, 45  
 qs, 45  
 lsgcfd  
 lsgc.c, 132  
 lspmac.c, 133  
   \_lspmac\_motor\_init, 146  
 alignx, 212  
 aligny, 212  
 alignz, 212  
 anal, 212  
 apery, 212  
 aperz, 212  
 arm\_parked, 212  
 blight, 212  
 blight\_down, 212  
 blight\_f, 213  
 blight\_ud, 213  
 blight\_up, 213  
 capy, 213  
 capz, 213  
 cenx, 213  
 ceny, 213  
 cleanstr, 147  
 cr\_cmd, 213  
 cryo, 213  
 cryo\_back, 214  
 cryo\_switch, 214  
 dbmem, 214  
 dbmemln, 214  
 dryer, 214  
 etel\_init\_ok, 214  
 etel\_on, 214  
 etel\_ready, 214  
 ethCmdOff, 214  
 ethCmdOn, 215  
 ethCmdQueue, 215  
 ethCmdReply, 215  
 flight, 215  
 flight\_f, 215  
 flight\_oo, 215  
 fluo, 215  
 fluor\_back, 215  
 fscint, 215  
 fshut, 216  
 gb\_cmd, 216  
 getivars, 216  
 getmvars, 216  
   hex\_dump, 147  
 hp\_air, 216  
 kappa, 216  
 LS\_PMAC\_STATE\_CR, 142  
 LS\_PMAC\_STATE\_GB, 142  
 LS\_PMAC\_STATE\_GMR, 142  
 LS\_PMAC\_STATE\_IDLE, 142  
 LS\_PMAC\_STATE\_RR, 143  
 LS\_PMAC\_STATE\_SC, 143  
 LS\_PMAC\_STATE\_WACK, 143  
 LS\_PMAC\_STATE\_WCR, 143  
 LS\_PMAC\_STATE\_WGB, 143  
 LSPMAC\_MAX\_MOTORS, 143  
 LSPMAC\_PRESET\_REGEX, 143  
 lp\_air, 216  
 ls\_pmac\_state, 216  
 lsConnect, 148  
 lspmac\_Error, 153  
 lspmac\_GetAllIVars, 167  
 lspmac\_GetAllIVarsCB, 167  
 lspmac\_GetAllMVars, 167  
 lspmac\_GetAllMVarsCB, 167  
 lspmac\_GetShortReplyCB, 169  
 lspmac\_Getmem, 168  
 lspmac\_GetmemReplyCB, 168  
 lspmac\_Reset, 194  
 lspmac\_SendControlReplyPrintCB, 201  
 lspmac\_Service, 202  
 lspmac\_SockFlush, 206  
 lspmac\_SockGetmem, 206  
 lspmac\_SockSendControlCharPrint, 207  
 lspmac\_SockSendDPControlChar, 207  
 lspmac\_SockSendDPControlCharCB, 207  
 lspmac\_SockSendDPLine, 207  
 lspmac\_SockSendDPqueue, 208  
 lspmac\_SockSendline, 208  
 lspmac\_SockSendline\_nr, 209  
 lspmac\_abort, 149  
 lspmac\_ascii\_buffers, 216  
 lspmac\_ascii\_buffers\_mutex, 216  
 lspmac\_ascii\_buffers\_t, 146  
 lspmac\_ascii\_busy, 217  
 lspmac\_ascii\_mutex, 217  
 lspmac\_asciicmdCB, 149  
 lspmac\_backLight\_down\_cb, 149  
 lspmac\_backLight\_up\_cb, 149  
 lspmac\_bi\_init, 150  
 lspmac\_bis, 217  
 lspmac\_blight\_lut\_setup, 150  
 lspmac\_bo\_init, 151  
 lspmac\_bo\_read, 151  
 lspmac\_combined\_move\_t, 146  
 lspmac\_command\_done\_cb, 152  
 lspmac\_control\_char, 217  
 lspmac\_cryoSwitchChanged\_cb, 152  
 lspmac\_dac\_init, 152  
 lspmac\_dac\_read, 153  
 lspmac\_dpascii\_off, 217

lspmac\_dpascii\_on, 217  
lspmac\_dpascii\_queue, 217  
lspmac\_dpascii\_queue\_t, 146  
lspmac\_est\_move\_time, 154  
lspmac\_est\_move\_time\_wait, 159  
lspmac\_find\_motor\_by\_name, 160  
lspmac\_flight\_lut\_setup, 160  
lspmac\_fscint\_lut\_setup, 160  
lspmac\_fshut\_init, 161  
lspmac\_full\_card\_reset\_cb, 161  
lspmac\_get\_ascii, 161  
lspmac\_get\_ascii\_cb, 162  
lspmac\_get\_status, 163  
lspmac\_get\_status\_cb, 164  
lspmac\_getBIPosition, 168  
lspmac\_getPosition, 168  
lspmac\_home1\_queue, 169  
lspmac\_home2\_queue, 170  
lspmac\_init, 171  
lspmac\_jogabs\_queue, 175  
lspmac\_light\_zoom\_cb, 175  
lspmac\_lut, 176  
lspmac\_more\_ascii\_cb, 177  
lspmac\_motor\_init, 177  
lspmac\_motors, 217  
lspmac\_move\_or\_jog\_abs\_queue, 178  
lspmac\_move\_or\_jog\_preset\_queue, 181  
lspmac\_move\_preset\_queue, 181  
lspmac\_moveabs\_blight\_factor\_queue, 182  
lspmac\_moveabs\_bo\_queue, 182  
lspmac\_moveabs\_flight\_factor\_queue, 183  
lspmac\_moveabs\_frontlight\_oo\_queue, 183  
lspmac\_moveabs\_fshut\_queue, 183  
lspmac\_moveabs\_queue, 184  
lspmac\_moveabs\_timed\_queue, 184  
lspmac\_moveabs\_wait, 185  
lspmac\_movedac\_queue, 186  
lspmac\_movezoom\_queue, 187  
lspmac\_moving\_cond, 217  
lspmac\_moving\_flags, 217  
lspmac\_moving\_mutex, 218  
lspmac\_nbis, 218  
lspmac\_next\_state, 188  
lspmac\_nmotors, 218  
lspmac\_pmacmotor\_read, 189  
lspmac\_pop\_queue, 192  
lspmac\_pop\_reply, 193  
lspmac\_push\_queue, 193  
lspmac\_quitting\_cb, 194  
lspmac\_request\_control\_response\_cb, 194  
lspmac\_reset\_queue, 194  
lspmac\_rlut, 195  
lspmac\_run, 196  
lspmac\_running, 218  
lspmac\_scint\_dried\_cb, 197  
lspmac\_scint\_maybe\_move\_sample\_cb, 198  
lspmac\_scint\_maybe\_return\_sample\_cb, 198  
lspmac\_scint\_maybe\_turn\_off\_dryer\_cb, 198  
lspmac\_scint\_maybe\_turn\_on\_dryer\_cb, 199  
lspmac\_send\_command, 199  
lspmac\_sendcmd, 200  
lspmac\_sendcmd\_nocb, 201  
lspmac\_set\_motion\_flags, 204  
lspmac\_shutter\_cond, 218  
lspmac\_shutter\_has\_opened, 218  
lspmac\_shutter\_mutex, 218  
lspmac\_shutter\_read, 205  
lspmac\_shutter\_state, 218  
lspmac\_soft\_motor\_init, 209  
lspmac\_soft\_motor\_read, 209  
lspmac\_status\_last\_time, 218  
lspmac\_status\_time, 219  
lspmac\_test\_preset, 209  
lspmac\_video\_rotate, 210  
lspmac\_worker, 210  
lspmac\_zoom\_lut\_setup, 211  
md2\_status, 219  
md2\_status\_mutex, 219  
md2\_status\_t, 146  
minikappa\_ok, 219  
motors\_ht, 219  
now, 219  
omega, 219  
omega\_zero\_search, 219  
omega\_zero\_time, 219  
omega\_zero\_velocity, 220  
PMAC\_MIN\_CMD\_TIME, 144  
PMACPORT, 144  
phi, 220  
pmac\_cmd\_size, 144  
pmac\_error\_strs, 220  
pmac\_queue\_cond, 220  
pmac\_queue\_mutex, 220  
pmac\_thread, 220  
pmacfd, 221  
rr\_cmd, 221  
sample\_detected, 221  
scint, 221  
shutter\_open, 221  
smart\_mag\_err, 221  
smart\_mag\_off, 221  
smart\_mag\_on, 221  
smart\_mag\_oo, 221  
VR\_CTRL\_RESPONSE, 144  
VR\_DOWNLOAD, 144  
VR\_FWDOWNLOAD, 144  
VR\_IPADDRESS, 144  
VR\_PMAC\_FLUSH, 144  
VR\_PMAC\_GETBUFFER, 144  
VR\_PMAC\_GETLINE, 144  
VR\_PMAC\_GETMEM, 145  
VR\_PMAC\_GETRESPONSE, 145  
VR\_PMAC\_PORT, 145  
VR\_PMAC\_READREADY, 145  
VR\_PMAC\_SENDLINE, 145  
VR\_PMAC\_SETBIT, 145

VR\_PMAC\_SETBITS, 145  
 VR\_PMAC\_SETMEM, 145  
 VR\_PMAC\_WRITEBUFFER, 145  
 VR\_PMAC\_WRITEERROR, 145  
 VR\_UPLOAD, 145  
 zoom, 222

**Ispmac\_Error**  
 Ispmac.c, 153

**Ispmac\_GetAllIVars**  
 Ispmac.c, 167

**Ispmac\_GetAllIVarsCB**  
 Ispmac.c, 167

**Ispmac\_GetAllMVars**  
 Ispmac.c, 167

**Ispmac\_GetAllMVarsCB**  
 Ispmac.c, 167

**Ispmac\_GetShortReplyCB**  
 Ispmac.c, 169

**Ispmac\_Getmem**  
 Ispmac.c, 168

**Ispmac\_GetmemReplyCB**  
 Ispmac.c, 168

**Ispmac\_Reset**  
 Ispmac.c, 194

**Ispmac\_SendControlReplyPrintCB**  
 Ispmac.c, 201

**Ispmac\_Service**  
 Ispmac.c, 202

**Ispmac\_SockFlush**  
 Ispmac.c, 206

**Ispmac\_SockGetmem**  
 Ispmac.c, 206

**Ispmac\_SockSendControlCharPrint**  
 Ispmac.c, 207  
 pgpmac.h, 341

**Ispmac\_SockSendDPControlChar**  
 Ispmac.c, 207  
 pgpmac.h, 342

**Ispmac\_SockSendDPControlCharCB**  
 Ispmac.c, 207

**Ispmac\_SockSendDPLine**  
 Ispmac.c, 207  
 pgpmac.h, 342

**Ispmac\_SockSendDPqueue**  
 Ispmac.c, 208

**Ispmac\_SockSendLine**  
 Ispmac.c, 208  
 pgpmac.h, 342

**Ispmac\_SockSendLine\_nr**  
 Ispmac.c, 209

**Ispmac\_abort**  
 Ispmac.c, 149  
 pgpmac.h, 321

**Ispmac\_ascii\_buffers**  
 Ispmac.c, 216

**Ispmac\_ascii\_buffers\_mutex**  
 Ispmac.c, 216

**Ispmac\_ascii\_buffers\_struct**, 45

command\_buf, 45  
 command\_buf\_cc, 45  
 command\_str, 45  
 response\_buf, 46  
 response\_n, 46  
 response\_str, 46

**Ispmac\_ascii\_buffers\_t**  
 Ispmac.c, 146

**Ispmac\_ascii\_busy**  
 Ispmac.c, 217

**Ispmac\_ascii\_mutex**  
 Ispmac.c, 217

**Ispmac\_asciicmdCB**  
 Ispmac.c, 149

**Ispmac\_backLight\_down\_cb**  
 Ispmac.c, 149

**Ispmac\_backLight\_up\_cb**  
 Ispmac.c, 149

**Ispmac\_bi\_init**  
 Ispmac.c, 150

**Ispmac\_bi\_struct**, 46  
 changeEventOff, 47  
 changeEventOn, 47  
 first\_time, 47  
 mask, 47  
 mutex, 47  
 position, 47  
 previous, 47  
 ptr, 47

**Ispmac\_bi\_t**  
 pgpmac.h, 308

**Ispmac\_bis**  
 Ispmac.c, 217

**Ispmac\_blight\_lut\_setup**  
 Ispmac.c, 150

**Ispmac\_bo\_init**  
 Ispmac.c, 151

**Ispmac\_bo\_read**  
 Ispmac.c, 151

**Ispmac\_cmd\_queue\_struct**, 48  
 event, 48  
 no\_reply, 48  
 onResponse, 48  
 pcmd, 48  
 time\_sent, 48

**Ispmac\_combined\_move\_struct**, 49  
 axis, 49  
 coord\_num, 49  
 Delta, 49  
 moveme, 49

**Ispmac\_combined\_move\_t**  
 Ispmac.c, 146

**Ispmac\_command\_done\_cb**  
 Ispmac.c, 152

**Ispmac\_control\_char**  
 Ispmac.c, 217

**Ispmac\_cryoSwitchChanged\_cb**  
 Ispmac.c, 152

lspmac\_dac\_init  
    lspmac.c, 152  
lspmac\_dac\_read  
    lspmac.c, 153  
lspmac\_dpascii\_off  
    lspmac.c, 217  
lspmac\_dpascii\_on  
    lspmac.c, 217  
lspmac\_dpascii\_queue  
    lspmac.c, 217  
lspmac\_dpascii\_queue\_struct, 49  
    event, 50  
    pl, 50  
lspmac\_dpascii\_queue\_t  
    lspmac.c, 146  
lspmac\_est\_move\_time  
    lspmac.c, 154  
    pgpmac.h, 321  
lspmac\_est\_move\_time\_wait  
    lspmac.c, 159  
    pgpmac.h, 326  
lspmac\_find\_motor\_by\_name  
    lspmac.c, 160  
    pgpmac.h, 327  
lspmac\_flight\_lut\_setup  
    lspmac.c, 160  
lspmac\_fscint\_lut\_setup  
    lspmac.c, 160  
lspmac\_fshut\_init  
    lspmac.c, 161  
lspmac\_full\_card\_reset\_cb  
    lspmac.c, 161  
lspmac\_get\_ascii  
    lspmac.c, 161  
lspmac\_get\_ascii\_cb  
    lspmac.c, 162  
lspmac\_get\_status  
    lspmac.c, 163  
lspmac\_get\_status\_cb  
    lspmac.c, 164  
lspmac\_getBIPosition  
    lspmac.c, 168  
    pgpmac.h, 327  
lspmac\_getPosition  
    lspmac.c, 168  
    pgpmac.h, 327  
lspmac\_home1\_queue  
    lspmac.c, 169  
    pgpmac.h, 328  
lspmac\_home2\_queue  
    lspmac.c, 170  
    pgpmac.h, 329  
lspmac\_init  
    lspmac.c, 171  
    pgpmac.h, 329  
lspmac\_jogabs\_queue  
    lspmac.c, 175  
    pgpmac.h, 333  
lspmac\_light\_zoom\_cb  
    lspmac.c, 175  
lspmac\_lut  
    lspmac.c, 176  
lspmac\_more\_ascii\_cb  
    lspmac.c, 177  
lspmac\_motor\_init  
    lspmac.c, 177  
lspmac\_motor\_struct, 50  
    active, 52  
    active\_init, 52  
    actual\_pos\_cnts, 52  
    actual\_pos\_cnts\_p, 53  
    axis, 53  
    command\_sent, 53  
    cond, 53  
    coord\_num, 53  
    dac\_mvar, 53  
    home, 53  
    homing, 53  
    in\_position\_band, 53  
    inactive\_init, 54  
    jogAbs, 54  
    lut, 54  
    magic, 54  
    max\_accel, 54  
    max\_pos, 54  
    max\_speed, 54  
    min\_pos, 54  
    motion\_seen, 54  
    motor\_num, 55  
    moveAbs, 55  
    mutex, 55  
    name, 55  
    neg\_limit\_hit, 55  
    neutral\_pos, 55  
    nlut, 55  
    not\_done, 55  
    pos\_limit\_hit, 55  
    position, 56  
    pq, 56  
    precision, 56  
    printf\_fmt, 56  
    read, 56  
    read\_mask, 56  
    read\_ptr, 56  
    redis\_fmt, 56  
    redis\_position, 56  
    reported\_pg\_position, 57  
    reported\_position, 57  
    requested\_pos\_cnts, 57  
    requested\_position, 57  
    status1, 57  
    status1\_p, 57  
    status2, 57  
    status2\_p, 57  
    status\_str, 57  
    u2c, 58

unit, 58  
 update\_resolution, 58  
 win, 58  
 write\_fmt, 58  
**lspmac\_motor\_t**  
 pgpmac.h, 308  
**lspmac\_motors**  
 lspmac.c, 217  
 pgpmac.h, 360  
**lspmac\_move\_or\_jog\_abs\_queue**  
 lspmac.c, 178  
 pgpmac.h, 333  
**lspmac\_move\_or\_jog\_preset\_queue**  
 lspmac.c, 181  
 pgpmac.h, 336  
**lspmac\_move\_or\_jog\_queue**  
 pgpmac.h, 337  
**lspmac\_move\_preset\_queue**  
 lspmac.c, 181  
 pgpmac.h, 337  
**lspmac\_moveabs\_blight\_factor\_queue**  
 lspmac.c, 182  
**lspmac\_moveabs\_bo\_queue**  
 lspmac.c, 182  
**lspmac\_moveabs\_flight\_factor\_queue**  
 lspmac.c, 183  
**lspmac\_moveabs\_frontlight\_oo\_queue**  
 lspmac.c, 183  
**lspmac\_moveabs\_fshut\_queue**  
 lspmac.c, 183  
**lspmac\_moveabs\_queue**  
 lspmac.c, 184  
 pgpmac.h, 337  
**lspmac\_moveabs\_timed\_queue**  
 lspmac.c, 184  
**lspmac\_moveabs\_wait**  
 lspmac.c, 185  
 pgpmac.h, 337  
**lspmac\_movedac\_queue**  
 lspmac.c, 186  
**lspmac\_movezoom\_queue**  
 lspmac.c, 187  
**lspmac\_moving\_cond**  
 lspmac.c, 217  
 pgpmac.h, 360  
**lspmac\_moving\_flags**  
 lspmac.c, 217  
 pgpmac.h, 360  
**lspmac\_moving\_mutex**  
 lspmac.c, 218  
 pgpmac.h, 360  
**lspmac\_nbis**  
 lspmac.c, 218  
**lspmac\_next\_state**  
 lspmac.c, 188  
**lspmac\_nmotors**  
 lspmac.c, 218  
 pgpmac.h, 360

lspmac\_pmacmotor\_read  
 lspmac.c, 189  
**lspmac\_pop\_queue**  
 lspmac.c, 192  
**lspmac\_pop\_reply**  
 lspmac.c, 193  
**lspmac\_push\_queue**  
 lspmac.c, 193  
**lspmac\_quitting\_cb**  
 lspmac.c, 194  
**lspmac\_request\_control\_response\_cb**  
 lspmac.c, 194  
**lspmac\_reset\_queue**  
 lspmac.c, 194  
**lspmac\_rlut**  
 lspmac.c, 195  
**lspmac\_run**  
 lspmac.c, 196  
 pgpmac.h, 339  
**lspmac\_running**  
 lspmac.c, 218  
**lspmac\_scint\_dried\_cb**  
 lspmac.c, 197  
**lspmac\_scint\_maybe\_move\_sample\_cb**  
 lspmac.c, 198  
**lspmac\_scint\_maybe\_return\_sample\_cb**  
 lspmac.c, 198  
**lspmac\_scint\_maybe\_turn\_off\_dryer\_cb**  
 lspmac.c, 198  
**lspmac\_scint\_maybe\_turn\_on\_dryer\_cb**  
 lspmac.c, 199  
**lspmac\_send\_command**  
 lspmac.c, 199  
**lspmac\_sendcmd**  
 lspmac.c, 200  
**lspmac\_sendcmd\_noob**  
 lspmac.c, 201  
**lspmac\_set\_motion\_flags**  
 lspmac.c, 204  
 pgpmac.h, 340  
**lspmac\_shutter\_cond**  
 lspmac.c, 218  
 pgpmac.h, 360  
**lspmac\_shutter\_has\_opened**  
 lspmac.c, 218  
 pgpmac.h, 361  
**lspmac\_shutter\_mutex**  
 lspmac.c, 218  
 pgpmac.h, 361  
**lspmac\_shutter\_read**  
 lspmac.c, 205  
**lspmac\_shutter\_state**  
 lspmac.c, 218  
 pgpmac.h, 361  
**lspmac\_soft\_motor\_init**  
 lspmac.c, 209  
**lspmac\_soft\_motor\_read**  
 lspmac.c, 209

lspmac\_status\_last\_time  
  lspmac.c, 218  
lspmac\_status\_time  
  lspmac.c, 219  
lspmac\_test\_preset  
  lspmac.c, 209  
lspmac\_video\_rotate  
  lspmac.c, 210  
  pgpmac.h, 343  
lspmac\_worker  
  lspmac.c, 210  
lspmac\_zoom\_lut\_setup  
  lspmac.c, 211  
lsredis.c, 222  
  \_lsredis\_get\_obj, 225  
  \_lsredis\_set\_value, 226  
  lsredis\_addRead, 227  
  lsredis\_addWrite, 227  
  lsredis\_cleanup, 227  
  lsredis\_cmpnstr, 228  
  lsredis\_cmpstr, 228  
  lsredis\_cond, 244  
  lsredis\_config, 228  
  lsredis\_config\_cond, 244  
  lsredis\_config\_mutex, 244  
  lsredis\_configCB, 229  
  lsredis\_debugCB, 230  
  lsredis\_delRead, 231  
  lsredis\_delWrite, 231  
  lsredis\_fd\_service, 231  
  lsredis\_find\_preset, 231  
  lsredis\_find\_preset\_index\_by\_position, 232  
  lsredis\_get\_obj, 232  
  lsredis\_get\_or\_set\_d, 233  
  lsredis\_get\_or\_set\_l, 233  
  lsredis\_get\_string\_array, 234  
  lsredis\_getb, 234  
  lsredis\_getc, 234  
  lsredis\_getd, 234  
  lsredis\_getl, 235  
  lsredis\_getstr, 235  
  lsredis\_head, 244  
  lsredis\_hgetCB, 235  
  lsredis\_htab, 244  
  lsredis\_init, 236  
  lsredis\_key\_select\_regex, 244  
  lsredis\_keysCB, 237  
  lsredis\_load\_presets, 237  
  lsredis\_maybe\_add\_key, 238  
  lsredis\_mutex, 244  
  lsredis\_objs, 244  
  lsredis\_preset\_ht, 244  
  lsredis\_preset\_list, 245  
  lsredis\_preset\_list\_mutex, 245  
  lsredis\_preset\_list\_t, 225  
  lsredis\_preset\_max\_n, 245  
  lsredis\_preset\_n, 245  
  lsredis\_publisher, 245  
  lsredis\_reexec, 238  
  lsredis\_run, 239  
  lsredis\_running, 245  
  lsredis\_set\_preset, 239  
  lsredis\_set\_value, 240  
  lsredis\_setstr, 240  
  lsredis\_sig\_service, 241  
  lsredis\_subCB, 241  
  lsredis\_thread, 245  
  lsredis\_worker, 242  
  mutex\_initializer, 245  
  redisDisconnectCB, 244  
  roac, 245  
  rofd, 245  
  subac, 245  
  subfd, 245  
  wrac, 246  
  wrfd, 246  
  lsredis\_addRead  
    lsredis.c, 227  
  lsredis\_addWrite  
    lsredis.c, 227  
  lsredis\_cleanup  
    lsredis.c, 227  
  lsredis\_cmpnstr  
    lsredis.c, 228  
    pgpmac.h, 343  
  lsredis\_cmpstr  
    lsredis.c, 228  
    pgpmac.h, 343  
  lsredis\_cond  
    lsredis.c, 244  
    pgpmac.h, 361  
  lsredis\_config  
    lsredis.c, 228  
    pgpmac.h, 344  
  lsredis\_config\_cond  
    lsredis.c, 244  
  lsredis\_config\_mutex  
    lsredis.c, 244  
  lsredis\_configCB  
    lsredis.c, 229  
  lsredis\_debugCB  
    lsredis.c, 230  
  lsredis\_delRead  
    lsredis.c, 231  
  lsredis\_delWrite  
    lsredis.c, 231  
  lsredis\_fd\_service  
    lsredis.c, 231  
  lsredis\_find\_preset  
    lsredis.c, 231  
    pgpmac.h, 344  
  lsredis\_find\_preset\_index\_by\_position  
    lsredis.c, 232  
    pgpmac.h, 345  
  lsredis\_get\_obj  
    lsredis.c, 232

pgpmac.h, 345  
 lsredis\_get\_or\_set\_d  
     lsredis.c, 233  
 lsredis\_get\_or\_set\_l  
     lsredis.c, 233  
 lsredis\_get\_string\_array  
     lsredis.c, 234  
     pgpmac.h, 346  
 lsredis\_getb  
     lsredis.c, 234  
     pgpmac.h, 346  
 lsredis\_getc  
     lsredis.c, 234  
     pgpmac.h, 346  
 lsredis\_getd  
     lsredis.c, 234  
     pgpmac.h, 346  
 lsredis\_getl  
     lsredis.c, 235  
     pgpmac.h, 347  
 lsredis\_getstr  
     lsredis.c, 235  
     pgpmac.h, 347  
 lsredis\_head  
     lsredis.c, 244  
 lsredis\_hgetCB  
     lsredis.c, 235  
 lsredis\_htab  
     lsredis.c, 244  
 lsredis\_init  
     lsredis.c, 236  
     pgpmac.h, 347  
 lsredis\_key\_select\_regex  
     lsredis.c, 244  
 lsredis\_keysCB  
     lsredis.c, 237  
 lsredis\_load\_presets  
     lsredis.c, 237  
     pgpmac.h, 348  
 lsredis\_maybe\_add\_key  
     lsredis.c, 238  
 lsredis\_mutex  
     lsredis.c, 244  
     pgpmac.h, 361  
 lsredis\_obj\_struct, 58  
     avalue, 59  
     bvalue, 59  
     cond, 59  
     cvalue, 59  
     dvalue, 59  
     events\_name, 60  
     hits, 60  
     key, 60  
     lvalue, 60  
     mutex, 60  
     next, 60  
     valid, 60  
     value, 60  
         value\_length, 60  
         wait\_for\_me, 61  
 lsredis\_obj\_t  
     pgpmac.h, 308  
 lsredis\_objs  
     lsredis.c, 244  
 lsredis\_preset\_ht  
     lsredis.c, 244  
 lsredis\_preset\_list  
     lsredis.c, 245  
 lsredis\_preset\_list\_mutex  
     lsredis.c, 245  
 lsredis\_preset\_list\_struct, 61  
     index, 61  
     key, 61  
     name, 61  
     next, 61  
     position, 61  
 lsredis\_preset\_list\_t  
     lsredis.c, 225  
 lsredis\_preset\_max\_n  
     lsredis.c, 245  
 lsredis\_preset\_n  
     lsredis.c, 245  
 lsredis\_publisher  
     lsredis.c, 245  
 lsredis\_reexec  
     lsredis.c, 238  
     pgpmac.h, 349  
 lsredis\_run  
     lsredis.c, 239  
     pgpmac.h, 350  
 lsredis\_running  
     lsredis.c, 245  
     pgpmac.h, 361  
 lsredis\_set\_preset  
     lsredis.c, 239  
     pgpmac.h, 350  
 lsredis\_set\_value  
     lsredis.c, 240  
 lsredis\_setstr  
     lsredis.c, 240  
     pgpmac.h, 350  
 lsredis\_sig\_service  
     lsredis.c, 241  
 lsredis\_subCB  
     lsredis.c, 241  
 lsredis\_thread  
     lsredis.c, 245  
 lsredis\_worker  
     lsredis.c, 242  
 ltest.c, 246  
     ltest\_lspmac\_est\_move\_time, 246  
     ltest\_main, 247  
 ltest\_lspmac\_est\_move\_time  
     ltest.c, 246  
 ltest\_main  
     ltest.c, 247

pgpmac.h, 351  
lstim.c, 248  
    handler, 250  
    LSTIMER\_LIST\_LENGTH, 249  
    lstim\_active\_timers, 253  
    lstim\_cond, 253  
    lstim\_init, 250  
    lstim\_list, 254  
    lstim\_list\_t, 249  
    lstim\_mutex, 254  
    lstim\_run, 250  
    lstim\_set\_timer, 250  
    lstim\_thread, 254  
    lstim\_timerid, 254  
    lstim\_unset\_timer, 251  
    lstim\_worker, 251  
    new\_timer, 254  
    service\_timers, 252  
lstim\_active\_timers  
    lstim.c, 253  
lstim\_cond  
    lstim.c, 253  
lstim\_init  
    lstim.c, 250  
    pgpmac.h, 351  
lstim\_list  
    lstim.c, 254  
lstim\_list\_struct, 62  
    delay\_nsecs, 62  
    delay\_secs, 62  
    event, 63  
    init\_nsecs, 63  
    init\_secs, 63  
    last\_nsecs, 63  
    last\_secs, 63  
    ncalls, 63  
    next\_nsecs, 63  
    next\_secs, 63  
    shots, 63  
lstim\_list\_t  
    lstim.c, 249  
lstim\_mutex  
    lstim.c, 254  
lstim\_run  
    lstim.c, 250  
    pgpmac.h, 352  
lstim\_set\_timer  
    lstim.c, 250  
    pgpmac.h, 352  
lstim\_thread  
    lstim.c, 254  
lstim\_timerid  
    lstim.c, 254  
lstim\_unset\_timer  
    lstim.c, 251  
    pgpmac.h, 353  
lstim\_worker  
    lstim.c, 251  
lsupdate\_init  
    pgpmac.h, 353  
lsupdate\_run  
    pgpmac.h, 353  
ltime  
    lslogging\_queue\_struct, 23  
lut  
    lspmac\_motor\_struct, 54  
lvalue  
    lsredis\_obj\_struct, 60  
MD2CMDS\_CMD\_LENGTH  
    pgpmac.h, 308  
magic  
    lspmac\_motor\_struct, 54  
main  
    pgpmac.c, 292  
mask  
    lspmac\_bi\_struct, 47  
max\_accel  
    lspmac\_motor\_struct, 54  
max\_pos  
    lspmac\_motor\_struct, 54  
max\_speed  
    lspmac\_motor\_struct, 54  
md2\_status  
    lspmac.c, 219  
md2\_status\_mutex  
    lspmac.c, 219  
    pgpmac.h, 361  
md2\_status\_t  
    lspmac.c, 146  
md2StatusStruct, 64  
    acc11c\_1, 66  
    acc11c\_2, 66  
    acc11c\_3, 66  
    acc11c\_5, 66  
    acc11c\_6, 66  
    alignx\_act\_pos, 66  
    alignx\_status\_1, 66  
    alignx\_status\_2, 66  
    aligny\_act\_pos, 66  
    aligny\_status\_1, 66  
    aligny\_status\_2, 67  
    alignz\_act\_pos, 67  
    alignz\_status\_1, 67  
    alignz\_status\_2, 67  
    analyzer\_act\_pos, 67  
    analyzer\_status\_1, 67  
    analyzer\_status\_2, 67  
    aperturey\_act\_pos, 67  
    aperturey\_status\_1, 67  
    aperturey\_status\_2, 67  
    aperturez\_act\_pos, 67  
    aperturez\_status\_1, 67  
    aperturez\_status\_2, 68  
    back\_dac, 68  
    capy\_act\_pos, 68  
    capy\_status\_1, 68

capy\_status\_2, 68  
 capz\_act\_pos, 68  
 capz\_status\_1, 68  
 capz\_status\_2, 68  
 centerx\_act\_pos, 68  
 centerx\_status\_1, 68  
 centerx\_status\_2, 68  
 centery\_act\_pos, 68  
 centery\_status\_1, 69  
 centery\_status\_2, 69  
 dummy1, 69  
 dummy2, 69  
 dummy3, 69  
 dummy4, 69  
 dummy5, 69  
 dummy6, 69  
 dummy7, 69  
 dummy8, 69  
 dummy9, 69  
 dummyA, 69  
 dummyB, 70  
 front\_dac, 70  
 fs\_has\_opened, 70  
 fs\_has\_opened\_globally, 70  
 fs\_is\_open, 70  
 kappa\_act\_pos, 70  
 kappa\_status\_1, 70  
 kappa\_status\_2, 70  
 moving\_flags, 70  
 number\_passes, 70  
 omega\_act\_pos, 70  
 omega\_status\_1, 70  
 omega\_status\_2, 71  
 phi\_act\_pos, 71  
 phi\_status\_1, 71  
 phi\_status\_2, 71  
 phiscan, 71  
 scint\_act\_pos, 71  
 scint\_piezo, 71  
 scint\_status\_1, 71  
 scint\_status\_2, 71  
 zoom\_act\_pos, 71  
 zoom\_status\_1, 71  
 zoom\_status\_2, 71  
 md2cmds.c, 254  
     md2cmds\_abort, 257  
     md2cmds\_action\_queue, 257  
     md2cmds\_action\_wait, 258  
     md2cmds\_capz\_moving\_time, 287  
     md2cmds\_cmd, 287  
     md2cmds\_cmd\_kv\_t, 257  
     md2cmds\_cmd\_kvs, 287  
     md2cmds\_cmd\_pg\_kvs, 287  
     md2cmds\_cmd\_regex, 288  
     md2cmds\_collect, 258  
     md2cmds\_cond, 288  
     md2cmds\_coordsys\_1\_stopped\_cb, 263  
     md2cmds\_coordsys\_2\_stopped\_cb, 263  
     md2cmds\_coordsys\_3\_stopped\_cb, 263  
     md2cmds\_coordsys\_4\_stopped\_cb, 263  
     md2cmds\_coordsys\_5\_stopped\_cb, 263  
     md2cmds\_coordsys\_7\_stopped\_cb, 263  
     md2cmds\_hmap, 288  
     md2cmds\_home\_prep, 263  
     md2cmds\_home\_wait, 264  
     md2cmds\_homing\_cond, 288  
     md2cmds\_homing\_count, 288  
     md2cmds\_homing\_mutex, 288  
     md2cmds\_init, 264  
     md2cmds\_is\_moving, 265  
     md2cmds\_kappaphi\_move, 266  
     md2cmds\_maybe\_done\_homing\_cb, 266  
     md2cmds\_maybe\_done\_moving\_cb, 266  
     md2cmds\_maybe\_rotate\_done\_cb, 267  
     md2cmds\_md\_status\_code, 288  
     md2cmds\_move\_prep, 267  
     md2cmds\_move\_wait, 267  
     md2cmds\_moveAbs, 268  
     md2cmds\_moveRel, 269  
     md2cmds\_moving\_cond, 288  
     md2cmds\_moving\_count, 288  
     md2cmds\_moving\_mutex, 289  
     md2cmds\_moving\_queue\_wait, 289  
     md2cmds\_mutex, 289  
     md2cmds\_mvcenter\_move, 270  
     md2cmds\_organs\_move\_presets, 271  
     md2cmds\_phase\_beamLocation, 271  
     md2cmds\_phase\_center, 272  
     md2cmds\_phase\_change, 273  
     md2cmds\_phase\_dataCollection, 274  
     md2cmds\_phase\_manualMount, 274  
     md2cmds\_phase\_robotMount, 275  
     md2cmds\_phase\_safe, 276  
     md2cmds\_prep\_axis, 277  
     md2cmds\_push\_queue, 277  
     md2cmds\_rotate, 277  
     md2cmds\_rotate\_cb, 279  
     md2cmds\_run, 280  
     md2cmds\_run\_cmd, 280  
     md2cmds\_set, 281  
     md2cmds\_set\_scale\_cb, 282  
     md2cmds\_settransferpoint, 282  
     md2cmds\_test, 283  
     md2cmds\_thread, 289  
     md2cmds\_time\_capz\_cb, 283  
     md2cmds\_transfer, 284  
     md2cmds\_worker, 286  
     rotating, 289  
     md2cmds\_abort  
         md2cmds.c, 257  
     md2cmds\_action\_queue  
         md2cmds.c, 257  
     md2cmds\_action\_wait  
         md2cmds.c, 258  
     md2cmds\_capz\_moving\_time  
         md2cmds.c, 287

md2cmds\_cmd  
    md2cmds.c, 287  
    pgpmac.h, 361  
md2cmds\_cmd\_kv\_struct, 64  
    k, 64  
    v, 64  
md2cmds\_cmd\_kv\_t  
    md2cmds.c, 257  
md2cmds\_cmd\_kvs  
    md2cmds.c, 287  
md2cmds\_cmd\_pg\_kvs  
    md2cmds.c, 287  
md2cmds\_cmd\_regex  
    md2cmds.c, 288  
md2cmds\_collect  
    md2cmds.c, 258  
md2cmds\_cond  
    md2cmds.c, 288  
    pgpmac.h, 361  
md2cmds\_coorsys\_1\_stopped\_cb  
    md2cmds.c, 263  
md2cmds\_coorsys\_2\_stopped\_cb  
    md2cmds.c, 263  
md2cmds\_coorsys\_3\_stopped\_cb  
    md2cmds.c, 263  
md2cmds\_coorsys\_4\_stopped\_cb  
    md2cmds.c, 263  
md2cmds\_coorsys\_5\_stopped\_cb  
    md2cmds.c, 263  
md2cmds\_coorsys\_7\_stopped\_cb  
    md2cmds.c, 263  
md2cmds\_hmap  
    md2cmds.c, 288  
md2cmds\_home\_prep  
    md2cmds.c, 263  
md2cmds\_home\_wait  
    md2cmds.c, 264  
md2cmds\_homing\_cond  
    md2cmds.c, 288  
md2cmds\_homing\_count  
    md2cmds.c, 288  
md2cmds\_homing\_mutex  
    md2cmds.c, 288  
md2cmds\_init  
    md2cmds.c, 264  
    pgpmac.h, 353  
md2cmds\_is\_moving  
    md2cmds.c, 265  
md2cmds\_kappaphi\_move  
    md2cmds.c, 266  
md2cmds\_maybe\_done\_homing\_cb  
    md2cmds.c, 266  
md2cmds\_maybe\_done\_moving\_cb  
    md2cmds.c, 266  
md2cmds\_maybe\_rotate\_done\_cb  
    md2cmds.c, 267  
md2cmds\_md\_status\_code  
    md2cmds.c, 288  
                pgpmac.h, 361  
md2cmds\_move\_prep  
    md2cmds.c, 267  
md2cmds\_move\_wait  
    md2cmds.c, 267  
md2cmds\_moveAbs  
    md2cmds.c, 268  
md2cmds\_moveRel  
    md2cmds.c, 269  
md2cmds\_moving\_cond  
    md2cmds.c, 288  
md2cmds\_moving\_count  
    md2cmds.c, 288  
md2cmds\_moving\_mutex  
    md2cmds.c, 289  
md2cmds\_moving\_queue\_wait  
    md2cmds.c, 289  
md2cmds\_mutex  
    md2cmds.c, 289  
    pgpmac.h, 362  
md2cmds\_mvcenter\_move  
    md2cmds.c, 270  
md2cmds\_organs\_move\_presets  
    md2cmds.c, 271  
md2cmds\_pg\_cond  
    pgpmac.h, 362  
md2cmds\_pg\_mutex  
    pgpmac.h, 362  
md2cmds\_phase\_beamLocation  
    md2cmds.c, 271  
md2cmds\_phase\_center  
    md2cmds.c, 272  
md2cmds\_phase\_change  
    md2cmds.c, 273  
md2cmds\_phase\_dataCollection  
    md2cmds.c, 274  
md2cmds\_phase\_manualMount  
    md2cmds.c, 274  
md2cmds\_phase\_robotMount  
    md2cmds.c, 275  
md2cmds\_phase\_safe  
    md2cmds.c, 276  
md2cmds\_prep\_axis  
    md2cmds.c, 277  
md2cmds\_push\_queue  
    md2cmds.c, 277  
    pgpmac.h, 354  
md2cmds\_rotate  
    md2cmds.c, 277  
md2cmds\_rotate\_cb  
    md2cmds.c, 279  
md2cmds\_run  
    md2cmds.c, 280  
    pgpmac.h, 354  
md2cmds\_run\_cmd  
    md2cmds.c, 280  
md2cmds\_set  
    md2cmds.c, 281

md2cmds\_set\_scale\_cb  
     md2cmds.c, 282  
 md2cmds\_settransferpoint  
     md2cmds.c, 282  
 md2cmds\_test  
     md2cmds.c, 283  
 md2cmds\_thread  
     md2cmds.c, 289  
 md2cmds\_time\_capz\_cb  
     md2cmds.c, 283  
 md2cmds\_transfer  
     md2cmds.c, 284  
 md2cmds\_worker  
     md2cmds.c, 286  
 min\_pos  
     lspmac\_motor\_struct, 54  
 minikappa\_ok  
     lspmac.c, 219  
     pgpmac.h, 362  
 mk\_active\_init  
     mk\_pgpmac\_redis, 12  
 mk\_home  
     mk\_pgpmac\_redis, 12  
 mk\_inactive\_init  
     mk\_pgpmac\_redis, 13  
 mk\_pgpmac\_redis, 11  
     active\_simulation, 12  
     asis, 12  
     b, 13  
     bi\_list, 13  
     configs, 13  
     f, 14  
     fnc, 14  
     hard\_ini, 14  
     hard\_ini\_fields, 14  
     head, 14  
     hi, 14  
     i, 14  
     mk\_active\_init, 12  
     mk\_home, 12  
     mk\_inactive\_init, 13  
     motor\_dict, 14  
     motor\_field\_lists, 14  
     motor\_num, 14  
     motor\_presets, 15  
     p, 15  
     pi, 15  
     plcc2\_dict, 15  
     plcc2\_file, 15  
     ppos, 15  
     pref\_ini, 15  
     v, 16  
     x, 16  
     xlate, 16  
     y, 16  
     zoom\_settings, 16  
 mk\_pgpmac\_redis.py, 289  
 motion\_seen  
     lspmac\_motor\_struct, 54  
 motor\_dict  
     mk\_pgpmac\_redis, 14  
 motor\_field\_lists  
     mk\_pgpmac\_redis, 14  
 motor\_num  
     lspmac\_motor\_struct, 55  
     mk\_pgpmac\_redis, 14  
 motor\_presets  
     mk\_pgpmac\_redis, 15  
 motors\_ht  
     lspmac.c, 219  
 moveAbs  
     lspmac\_motor\_struct, 55  
 moveme  
     lspmac\_combined\_move\_struct, 49  
 moving\_flags  
     md2StatusStruct, 70  
 mutex  
     lspg\_demandairrights\_struct, 23  
     lspg\_getcenter\_struct, 25  
     lspg\_getcurrentsampleid\_struct, 27  
     lspg\_lock\_detector\_struct, 28  
     lspg\_lock\_diffractometer\_struct, 28  
     lspg\_nextsample\_struct, 29  
     lspg\_nextshot\_struct, 39  
     lspg\_seq\_run\_prep\_struct, 41  
     lspg\_starttransfer\_struct, 42  
     lspg\_wait\_for\_detector\_struct, 43  
     lspg\_waitcryo\_struct, 44  
     lspmac\_bi\_struct, 47  
     lspmac\_motor\_struct, 55  
     lsredis\_obj\_struct, 60  
 mutex\_initializer  
     lsredis.c, 245  
 name  
     lspmac\_motor\_struct, 55  
     lsredis\_preset\_list\_struct, 61  
 ncalls  
     ltimer\_list\_struct, 63  
 ncurses\_mutex  
     pgpmac.c, 297  
     pgpmac.h, 362  
 neg\_limit\_hit  
     lspmac\_motor\_struct, 55  
 neutral\_pos  
     lspmac\_motor\_struct, 55  
 new\_timer  
     ltimer.c, 254  
 new\_value\_ready  
     lspg\_demandairrights\_struct, 23  
     lspg\_getcenter\_struct, 26  
     lspg\_getcurrentsampleid\_struct, 27  
     lspg\_lock\_detector\_struct, 28  
     lspg\_lock\_diffractometer\_struct, 28  
     lspg\_nextsample\_struct, 29  
     lspg\_nextshot\_struct, 39  
     lspg\_seq\_run\_prep\_struct, 41

lspg\_starttransfer\_struct, 42  
lspg\_wait\_for\_detector\_struct, 43  
lspg\_waitcryo\_struct, 44  
next  
  lsevents\_callbacks\_struct, 20  
  lsevents\_event\_names\_struct, 20  
  lsevents\_listener\_struct, 21  
  lsredis\_obj\_struct, 60  
  lsredis\_preset\_list\_struct, 61  
next\_nsecs  
  lstimer\_list\_struct, 63  
next\_secs  
  lstimer\_list\_struct, 63  
nextsample  
  lspg\_nextsample\_struct, 29  
nextsample\_isnull  
  lspg\_nextsample\_struct, 30  
nlut  
  lspmac\_motor\_struct, 55  
no\_reply  
  lspmac\_cmd\_queue\_struct, 48  
no\_rows\_returned  
  lspg\_getcenter\_struct, 26  
  lspg\_getcurrentsampleid\_struct, 27  
  lspg\_nextsample\_struct, 30  
  lspg\_nextshot\_struct, 39  
  lspg\_starttransfer\_struct, 42  
not\_done  
  lspmac\_motor\_struct, 55  
now  
  lspg.c, 132  
  lspmac.c, 219  
number\_passes  
  md2StatusStruct, 70  
  
omega  
  lspmac.c, 219  
  pgpmac.h, 362  
omega\_act\_pos  
  md2StatusStruct, 70  
omega\_status\_1  
  md2StatusStruct, 70  
omega\_status\_2  
  md2StatusStruct, 71  
omega\_zero\_search  
  lspmac.c, 219  
omega\_zero\_time  
  lspmac.c, 219  
  pgpmac.h, 362  
omega\_zero\_velocity  
  lspmac.c, 220  
onResponse  
  lspgQueryQueueStruct, 45  
  lspmac\_cmd\_queue\_struct, 48  
options  
  iniParser::iniParser, 18  
  
p  
  mk\_pgpmac\_redis, 15  
  PMAC\_MIN\_CMD\_TIME  
    lspmac.c, 144  
  PMACPORT  
    lspmac.c, 144  
  pcmd  
    lspmac\_cmd\_queue\_struct, 48  
  pgpmac.c, 290  
    doomsday\_count, 297  
    doomsday\_mutex, 297  
    main, 292  
    ncurses\_mutex, 297  
    pgpmac\_printf, 294  
    pgpmac\_quit\_cb, 294  
    pgpmac\_request\_stay\_of\_execution, 294  
    pgpmac\_use\_autoscint, 297  
    pgpmac\_use\_pg, 298  
    running, 298  
    stdinService, 295  
    stdinfd, 298  
    term\_input, 298  
    term\_output, 298  
    term\_status, 298  
    term\_status2, 298  
  pgpmac.h, 298  
    \_GNU\_SOURCE, 307  
    \_lsredis\_get\_obj, 309  
    alignx, 356  
    aligny, 356  
    alignz, 356  
    anal, 356  
    apery, 356  
    aperz, 356  
    arm\_parked, 356  
    blight, 356  
    blight\_down, 356  
    blight\_f, 356  
    blight\_ud, 357  
    blight\_up, 357  
    capy, 357  
    capz, 357  
    cenx, 357  
    ceny, 357  
    cryo, 357  
    cryo\_back, 357  
    cryo\_switch, 357  
    dryer, 358  
    etel\_init\_ok, 358  
    etel\_on, 358  
    etel\_ready, 358  
    flight, 358  
    flight\_f, 358  
    flight\_oo, 358  
    fluo, 358  
    fluor\_back, 358  
    fscint, 359  
    fshut, 359  
    hp\_air, 359  
    kappa, 359

LSPMAC\_MAGIC\_NUMBER, 307  
 lp\_air, 359  
 lsevents\_add\_listener, 310  
 lsevents\_init, 311  
 lsevents\_preregister\_event, 311  
 lsevents\_remove\_listener, 311  
 lsevents\_run, 312  
 lsevents\_send\_event, 313  
 lslogging\_init, 313  
 lslogging\_log\_message, 313  
 lslogging\_run, 314  
 lspg\_array2ptrs, 314  
 lspg\_demandairrights, 359  
 lspg\_demandairrights\_all, 316  
 lspg\_demandairrights\_t, 308  
 lspg\_getcenter, 359  
 lspg\_getcenter\_call, 316  
 lspg\_getcenter\_done, 316  
 lspg\_getcenter\_t, 308  
 lspg\_getcenter\_wait, 316  
 lspg\_getcurrentsampleid, 359  
 lspg\_getcurrentsampleid\_t, 308  
 lspg\_getcurrentsampleid\_wait\_for\_id, 316  
 lspg\_init, 317  
 lspg\_nextsample, 359  
 lspg\_nextsample\_all, 317  
 lspg\_nextsample\_t, 308  
 lspg\_nextshot, 360  
 lspg\_nextshot\_call, 317  
 lspg\_nextshot\_done, 318  
 lspg\_nextshot\_t, 308  
 lspg\_nextshot\_wait, 318  
 lspg\_query\_push, 318  
 lspg\_query\_queue\_t, 308  
 lspg\_run, 319  
 lspg\_seq\_run\_prep\_all, 319  
 lspg\_starttransfer, 360  
 lspg\_starttransfer\_call, 319  
 lspg\_starttransfer\_done, 320  
 lspg\_starttransfer\_t, 308  
 lspg\_starttransfer\_wait, 320  
 lspg\_waitcryo, 360  
 lspg\_waitcryo\_all, 320  
 lspg\_waitcryo\_cb, 320  
 lspg\_waitcryo\_t, 308  
 lspg\_zoom\_lut\_call, 320  
 lspmac\_SockSendControlCharPrint, 341  
 lspmac\_SockSendDPControlChar, 342  
 lspmac\_SockSendDPLine, 342  
 lspmac\_SockSendline, 342  
 lspmac\_abort, 321  
 lspmac\_bi\_t, 308  
 lspmac\_est\_move\_time, 321  
 lspmac\_est\_move\_time\_wait, 326  
 lspmac\_find\_motor\_by\_name, 327  
 lspmac\_getBIPosition, 327  
 lspmac\_getPosition, 327  
 lspmac\_home1\_queue, 328  
 lspmac\_home2\_queue, 329  
 lspmac\_init, 329  
 lspmac\_jogabs\_queue, 333  
 lspmac\_motor\_t, 308  
 lspmac\_motors, 360  
 lspmac\_move\_or\_jog\_abs\_queue, 333  
 lspmac\_move\_or\_jog\_preset\_queue, 336  
 lspmac\_move\_or\_jog\_queue, 337  
 lspmac\_move\_preset\_queue, 337  
 lspmac\_moveabs\_queue, 337  
 lspmac\_moveabs\_wait, 337  
 lspmac\_moving\_cond, 360  
 lspmac\_moving\_flags, 360  
 lspmac\_moving\_mutex, 360  
 lspmac\_nmotors, 360  
 lspmac\_run, 339  
 lspmac\_set\_motion\_flags, 340  
 lspmac\_shutter\_cond, 360  
 lspmac\_shutter\_has\_opened, 361  
 lspmac\_shutter\_mutex, 361  
 lspmac\_shutter\_state, 361  
 lspmac\_video\_rotate, 343  
 lsredis\_cmpnstr, 343  
 lsredis\_cmpstr, 343  
 lsredis\_cond, 361  
 lsredis\_config, 344  
 lsredis\_find\_preset, 344  
 lsredis\_find\_preset\_index\_by\_position, 345  
 lsredis\_get\_obj, 345  
 lsredis\_get\_string\_array, 346  
 lsredis\_getb, 346  
 lsredis\_getc, 346  
 lsredis\_getd, 346  
 lsredis\_getl, 347  
 lsredis\_getstr, 347  
 lsredis\_init, 347  
 lsredis\_load\_presets, 348  
 lsredis\_mutex, 361  
 lsredis\_obj\_t, 308  
 lsredis\_reexec, 349  
 lsredis\_run, 350  
 lsredis\_running, 361  
 lsredis\_set\_preset, 350  
 lsredis\_setstr, 350  
 ltest\_main, 351  
 ltimer\_init, 351  
 ltimer\_run, 352  
 ltimer\_set\_timer, 352  
 ltimer\_unset\_timer, 353  
 lsupdate\_init, 353  
 lsupdate\_run, 353  
 MD2CMDS\_CMD\_LENGTH, 308  
 md2\_status\_mutex, 361  
 md2cmds\_cmd, 361  
 md2cmds\_cond, 361  
 md2cmds\_init, 353  
 md2cmds\_md\_status\_code, 361  
 md2cmds\_mutex, 362

md2cmds\_pg\_cond, 362  
md2cmds\_pg\_mutex, 362  
md2cmds\_push\_queue, 354  
md2cmds\_run, 354  
minikappa\_ok, 362  
ncurses\_mutex, 362  
omega, 362  
omega\_zero\_time, 362  
pgpmac\_printf, 355  
pgpmac\_request\_stay\_of\_execution, 355  
pgpmac\_use\_autoscint, 362  
pgpmac\_use\_pg, 362  
phi, 362  
pmac\_cmd\_queue\_t, 309  
pmac\_cmd\_t, 309  
pmac\_queue\_cond, 362  
pmac\_queue\_mutex, 363  
PmacSockSendline, 355  
sample\_detected, 363  
scint, 363  
shutter\_open, 363  
smart\_mag\_err, 363  
smart\_mag\_off, 363  
smart\_mag\_on, 363  
smart\_mag\_oo, 363  
term\_input, 363  
term\_output, 364  
term\_status, 364  
term\_status2, 364  
zoom, 364  
pgpmac\_printf  
    pgpmac.c, 294  
    pgpmac.h, 355  
pgpmac\_quit\_cb  
    pgpmac.c, 294  
pgpmac\_request\_stay\_of\_execution  
    pgpmac.c, 294  
    pgpmac.h, 355  
pgpmac\_use\_autoscint  
    pgpmac.c, 297  
    pgpmac.h, 362  
pgpmac\_use\_pg  
    pgpmac.c, 298  
    pgpmac.h, 362  
phi  
    lspmac.c, 220  
    pgpmac.h, 362  
phi\_act\_pos  
    md2StatusStruct, 71  
phi\_status\_1  
    md2StatusStruct, 71  
phi\_status\_2  
    md2StatusStruct, 71  
phiscan  
    md2StatusStruct, 71  
pi  
    mk\_pgpmac\_redis, 15  
pl  
    lspmac\_dpascii\_queue\_struct, 50  
plcc2\_dict  
    mk\_pgpmac\_redis, 15  
plcc2\_file  
    mk\_pgpmac\_redis, 15  
pmac\_cmd\_queue\_t  
    pgpmac.h, 309  
pmac\_cmd\_size  
    lspmac.c, 144  
pmac\_cmd\_t  
    pgpmac.h, 309  
pmac\_error\_strs  
    lspmac.c, 220  
pmac\_queue\_cond  
    lspmac.c, 220  
    pgpmac.h, 362  
pmac\_queue\_mutex  
    lspmac.c, 220  
    pgpmac.h, 363  
pmac\_thread  
    lspmac.c, 220  
PmacSockSendline  
    pgpmac.h, 355  
pmacfd  
    lspmac.c, 221  
pos\_limit\_hit  
    lspmac\_motor\_struct, 55  
position  
    lspmac\_bi\_struct, 47  
    lspmac\_motor\_struct, 56  
    lsredis\_preset\_list\_struct, 61  
ppos  
    mk\_pgpmac\_redis, 15  
pq  
    lspmac\_motor\_struct, 56  
precision  
    lspmac\_motor\_struct, 56  
pref\_ini  
    mk\_pgpmac\_redis, 15  
previous  
    lspmac\_bi\_struct, 47  
printf\_fmt  
    lspmac\_motor\_struct, 56  
ptr  
    lspmac\_bi\_struct, 47  
q  
    lspg.c, 132  
qs  
    lspgQueryQueueStruct, 45  
raw\_regexp  
    lsevents\_listener\_struct, 21  
re  
    lsevents\_listener\_struct, 21  
read  
    iniParser::iniParser, 18  
    lspmac\_motor\_struct, 56  
read\_mask

lspmac\_motor\_struct, 56  
 read\_ptr  
     lspmac\_motor\_struct, 56  
 redis\_fmt  
     lspmac\_motor\_struct, 56  
 redis\_position  
     lspmac\_motor\_struct, 56  
 redisDisconnectCB  
     lsredis.c, 244  
 reported\_pg\_position  
     lspmac\_motor\_struct, 57  
 reported\_position  
     lspmac\_motor\_struct, 57  
 Request  
     tagEthernetCmd, 72  
 RequestType  
     tagEthernetCmd, 72  
 requested\_pos\_cnts  
     lspmac\_motor\_struct, 57  
 requested\_position  
     lspmac\_motor\_struct, 57  
 response\_buf  
     lspmac\_ascii\_buffers\_struct, 46  
 response\_n  
     lspmac\_ascii\_buffers\_struct, 46  
 response\_str  
     lspmac\_ascii\_buffers\_struct, 46  
 roac  
     lsredis.c, 245  
 rofd  
     lsredis.c, 245  
 rotating  
     md2cmds.c, 289  
 rr\_cmd  
     lspmac.c, 221  
 running  
     pgpmac.c, 298  
 sample\_detected  
     lspmac.c, 221  
     pgpmac.h, 363  
 scint  
     lspmac.c, 221  
     pgpmac.h, 363  
 scint\_act\_pos  
     md2StatusStruct, 71  
 scint\_piezo  
     md2StatusStruct, 71  
 scint\_status\_1  
     md2StatusStruct, 71  
 scint\_status\_2  
     md2StatusStruct, 71  
 sd  
     iniParser::iniParser, 19  
 sections  
     iniParser::iniParser, 19  
 service\_timers  
     ltimer.c, 252  
 sfn  
     lspg\_nextshot\_struct, 39  
 sfn\_isnull  
     lspg\_nextshot\_struct, 39  
 shots  
     ltimer\_list\_struct, 63  
 shutter\_open  
     lspmac.c, 221  
     pgpmac.h, 363  
 sindex  
     lspg\_nextshot\_struct, 39  
 sindex2  
     lspg\_nextshot\_struct, 39  
 sindex2\_isnull  
     lspg\_nextshot\_struct, 40  
 sindex\_isnull  
     lspg\_nextshot\_struct, 40  
 skey  
     lspg\_nextshot\_struct, 40  
 skey\_isnull  
     lspg\_nextshot\_struct, 40  
 smart\_mag\_err  
     lspmac.c, 221  
     pgpmac.h, 363  
 smart\_mag\_off  
     lspmac.c, 221  
     pgpmac.h, 363  
 smart\_mag\_on  
     lspmac.c, 221  
     pgpmac.h, 363  
 smart\_mag\_oo  
     lspmac.c, 221  
     pgpmac.h, 363  
 sstart  
     lspg\_nextshot\_struct, 40  
 sstart2  
     lspg\_nextshot\_struct, 40  
 sstart2\_isnull  
     lspg\_nextshot\_struct, 40  
 sstart\_isnull  
     lspg\_nextshot\_struct, 40  
 starttransfer  
     lspg\_starttransfer\_struct, 42  
 status1  
     lspmac\_motor\_struct, 57  
 status1\_p  
     lspmac\_motor\_struct, 57  
 status2  
     lspmac\_motor\_struct, 57  
 status2\_p  
     lspmac\_motor\_struct, 57  
 status\_str  
     lspmac\_motor\_struct, 57  
 stdInService  
     pgpmac.c, 295  
 stdInfdA  
     pgpmac.c, 298  
 stype  
     lspg\_nextshot\_struct, 40

stype2  
  lspg\_nextshot\_struct, 40

stype2\_isnull  
  lspg\_nextshot\_struct, 41

stype\_isnull  
  lspg\_nextshot\_struct, 41

subac  
  lsredis.c, 245

subfd  
  lsredis.c, 245

tagEthernetCmd, 72  
  bData, 72  
  Request, 72  
  RequestType, 72  
  wIndex, 73  
  wLength, 73  
  wValue, 73

term\_input  
  pgpmac.c, 298  
  pgpmac.h, 363

term\_output  
  pgpmac.c, 298  
  pgpmac.h, 364

term\_status  
  pgpmac.c, 298  
  pgpmac.h, 364

term\_status2  
  pgpmac.c, 298  
  pgpmac.h, 364

time\_sent  
  lspmac\_cmd\_queue\_struct, 48

u2c  
  lspmac\_motor\_struct, 58

unit  
  lspmac\_motor\_struct, 58

update\_resolution  
  lspmac\_motor\_struct, 58

v  
  md2cmds\_cmd\_kv\_struct, 64  
  mk\_pgpmac\_redis, 16

VR\_CTRL\_RESPONSE  
  lspmac.c, 144

VR\_DOWNLOAD  
  lspmac.c, 144

VR\_FDOWNLOAD  
  lspmac.c, 144

VR\_IPADDRESS  
  lspmac.c, 144

VR\_PMAC\_FLUSH  
  lspmac.c, 144

VR\_PMAC\_GETBUFFER  
  lspmac.c, 144

VR\_PMAC\_GETLINE  
  lspmac.c, 144

VR\_PMAC\_GETMEM  
  lspmac.c, 145

VR\_PMAC\_GETRESPONSE  
  lspmac.c, 145

VR\_PMAC\_PORT  
  lspmac.c, 145

VR\_PMAC\_READREADY  
  lspmac.c, 145

VR\_PMAC\_SENDCTRLCHAR  
  lspmac.c, 145

VR\_PMAC\_SENDLINE  
  lspmac.c, 145

VR\_PMAC\_SETBIT  
  lspmac.c, 145

VR\_PMAC\_SETBITS  
  lspmac.c, 145

VR\_PMAC\_SETMEM  
  lspmac.c, 145

VR\_PMAC\_WRITEBUFFER  
  lspmac.c, 145

VR\_PMAC\_WRITEERROR  
  lspmac.c, 145

VR\_UPLOAD  
  lspmac.c, 145

valid  
  lsredis\_obj\_struct, 60

value  
  lsredis\_obj\_struct, 60

value\_length  
  lsredis\_obj\_struct, 60

wIndex  
  tagEthernetCmd, 73

wLength  
  tagEthernetCmd, 73

wValue  
  tagEthernetCmd, 73

wait\_for\_me  
  lsredis\_obj\_struct, 61

win  
  lspmac\_motor\_struct, 58

wrac  
  lsredis.c, 246

wrfd  
  lsredis.c, 246

write\_fmt  
  lspmac\_motor\_struct, 58

x  
  mk\_pgpmac\_redis, 16

xlate  
  mk\_pgpmac\_redis, 16

y  
  mk\_pgpmac\_redis, 16

zoom  
  lspg\_getcenter\_struct, 26  
  lspmac.c, 222  
  pgpmac.h, 364

zoom\_act\_pos

md2StatusStruct, [71](#)  
zoom\_isnull  
  lspg\_getcenter\_struct, [26](#)  
zoom\_settings  
  mk\_pgpmac\_redis, [16](#)  
zoom\_status\_1  
  md2StatusStruct, [71](#)  
zoom\_status\_2  
  md2StatusStruct, [71](#)