

LS-CAT PGPMAC

Generated by Doxygen 1.8.2

Tue Jan 15 2013 16:53:10



# 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	Variable Documentation . . . . .	12
5.2.1.1	b . . . . .	12
5.2.1.2	bi_list . . . . .	12
5.2.1.3	f . . . . .	12
5.2.1.4	fnc . . . . .	12
5.2.1.5	hard_ini . . . . .	12
5.2.1.6	hard_ini_fields . . . . .	12
5.2.1.7	head . . . . .	12
5.2.1.8	hi . . . . .	12
5.2.1.9	i . . . . .	13
5.2.1.10	motor_dict . . . . .	13
5.2.1.11	motor_field_lists . . . . .	13
5.2.1.12	motor_presets . . . . .	13
5.2.1.13	p . . . . .	13
5.2.1.14	pi . . . . .	13
5.2.1.15	ppos . . . . .	13
5.2.1.16	pref_ini . . . . .	13

5.2.1.17	v	13
5.2.1.18	x	13
5.2.1.19	y	13
5.2.1.20	zoom_settings	13
<b>6</b>	<b>Data Structure Documentation</b>	<b>15</b>
6.1	iniParser.iniParser Class Reference	15
6.1.1	Detailed Description	15
6.1.2	Constructor & Destructor Documentation	16
6.1.2.1	__init__	16
6.1.3	Member Function Documentation	16
6.1.3.1	get	16
6.1.3.2	has_option	16
6.1.3.3	has_section	16
6.1.3.4	options	16
6.1.3.5	read	17
6.1.3.6	sections	17
6.1.4	Field Documentation	17
6.1.4.1	f	17
6.1.4.2	sd	17
6.2	Isevents_listener_struct Struct Reference	17
6.2.1	Detailed Description	18
6.2.2	Field Documentation	18
6.2.2.1	cb	18
6.2.2.2	next	18
6.2.2.3	raw_regexp	18
6.2.2.4	re	18
6.3	Isevents_queue_struct Struct Reference	18
6.3.1	Detailed Description	19
6.3.2	Field Documentation	19
6.3.2.1	evp	19
6.4	Islogging_queue_struct Struct Reference	19
6.4.1	Detailed Description	19
6.4.2	Field Documentation	19
6.4.2.1	lmsg	19
6.4.2.2	ltime	20
6.5	Ispg_demandairrights_struct Struct Reference	20
6.5.1	Detailed Description	20
6.5.2	Field Documentation	20
6.5.2.1	cond	20

6.5.2.2	mutex	20
6.5.2.3	new_value_ready	20
6.6	lspg_getcenter_struct Struct Reference	20
6.6.1	Detailed Description	21
6.6.2	Field Documentation	21
6.6.2.1	cond	21
6.6.2.2	dax	21
6.6.2.3	dax_isnull	21
6.6.2.4	day	22
6.6.2.5	day_isnull	22
6.6.2.6	daz	22
6.6.2.7	daz_isnull	22
6.6.2.8	dcx	22
6.6.2.9	dcx_isnull	22
6.6.2.10	dcy	22
6.6.2.11	dcy_isnull	22
6.6.2.12	mutex	22
6.6.2.13	new_value_ready	22
6.6.2.14	no_rows_returned	23
6.6.2.15	zoom	23
6.6.2.16	zoom_isnull	23
6.7	lspg_getcurrentsampleid_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	getcurrentsampleid	24
6.7.2.3	getcurrentsampleid_isnull	24
6.7.2.4	mutex	24
6.7.2.5	new_value_ready	24
6.7.2.6	no_rows_returned	24
6.8	lspg_lock_detector_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	mutex	25
6.8.2.3	new_value_ready	25
6.9	lspg_lock_diffractionmeter_struct Struct Reference	25
6.9.1	Detailed Description	25
6.9.2	Field Documentation	25
6.9.2.1	cond	25

6.9.2.2	mutex	25
6.9.2.3	new_value_ready	25
6.10	lspg_nextsample_struct Struct Reference	25
6.10.1	Detailed Description	26
6.10.2	Field Documentation	26
6.10.2.1	cond	26
6.10.2.2	mutex	26
6.10.2.3	new_value_ready	26
6.10.2.4	nextsample	26
6.10.2.5	nextsample_isnull	26
6.10.2.6	no_rows_returned	27
6.11	lspg_nextshot_struct Struct Reference	27
6.11.1	Detailed Description	29
6.11.2	Field Documentation	29
6.11.2.1	active	29
6.11.2.2	active2	30
6.11.2.3	active2_isnull	30
6.11.2.4	active_isnull	30
6.11.2.5	ax	30
6.11.2.6	ax2	30
6.11.2.7	ax2_isnull	30
6.11.2.8	ax_isnull	30
6.11.2.9	ay	30
6.11.2.10	ay2	30
6.11.2.11	ay2_isnull	30
6.11.2.12	ay_isnull	30
6.11.2.13	az	31
6.11.2.14	az2	31
6.11.2.15	az2_isnull	31
6.11.2.16	az_isnull	31
6.11.2.17	cond	31
6.11.2.18	cx	31
6.11.2.19	cx2	31
6.11.2.20	cx2_isnull	31
6.11.2.21	cx_isnull	31
6.11.2.22	cy	31
6.11.2.23	cy2	32
6.11.2.24	cy2_isnull	32
6.11.2.25	cy_isnull	32
6.11.2.26	dsdir	32

6.11.2.27 dsdir_isnull . . . . .	32
6.11.2.28 dsdist . . . . .	32
6.11.2.29 dsdist2 . . . . .	32
6.11.2.30 dsdist2_isnull . . . . .	32
6.11.2.31 dsdist_isnull . . . . .	32
6.11.2.32 dsexp . . . . .	32
6.11.2.33 dsexp2 . . . . .	33
6.11.2.34 dsexp2_isnull . . . . .	33
6.11.2.35 dsexp_isnull . . . . .	33
6.11.2.36 dshpid . . . . .	33
6.11.2.37 dshpid_isnull . . . . .	33
6.11.2.38 dskappa . . . . .	33
6.11.2.39 dskappa2 . . . . .	33
6.11.2.40 dskappa2_isnull . . . . .	33
6.11.2.41 dskappa_isnull . . . . .	33
6.11.2.42 dsnrg . . . . .	33
6.11.2.43 dsnrg2 . . . . .	34
6.11.2.44 dsnrg2_isnull . . . . .	34
6.11.2.45 dsnrg_isnull . . . . .	34
6.11.2.46 dsomega . . . . .	34
6.11.2.47 dsomega2 . . . . .	34
6.11.2.48 dsomega2_isnull . . . . .	34
6.11.2.49 dsomega_isnull . . . . .	34
6.11.2.50 dsoscaxis . . . . .	34
6.11.2.51 dsoscaxis2 . . . . .	34
6.11.2.52 dsoscaxis2_isnull . . . . .	34
6.11.2.53 dsoscaxis_isnull . . . . .	34
6.11.2.54 dsowidth . . . . .	35
6.11.2.55 dsowidth2 . . . . .	35
6.11.2.56 dsowidth2_isnull . . . . .	35
6.11.2.57 dsowidth_isnull . . . . .	35
6.11.2.58 dsphi . . . . .	35
6.11.2.59 dsphi2 . . . . .	35
6.11.2.60 dsphi2_isnull . . . . .	35
6.11.2.61 dsphi_isnull . . . . .	35
6.11.2.62 dspid . . . . .	35
6.11.2.63 dspid_isnull . . . . .	35
6.11.2.64 mutex . . . . .	36
6.11.2.65 new_value_ready . . . . .	36
6.11.2.66 no_rows_returned . . . . .	36

6.11.2.67	sfn	36
6.11.2.68	sfn_isnull	36
6.11.2.69	sindex	36
6.11.2.70	sindex2	36
6.11.2.71	sindex2_isnull	36
6.11.2.72	sindex_isnull	36
6.11.2.73	skey	36
6.11.2.74	skey_isnull	37
6.11.2.75	sstart	37
6.11.2.76	sstart2	37
6.11.2.77	sstart2_isnull	37
6.11.2.78	sstart_isnull	37
6.11.2.79	stype	37
6.11.2.80	stype2	37
6.11.2.81	stype2_isnull	37
6.11.2.82	stype_isnull	37
6.12	lspg_seq_run_prep_struct Struct Reference	37
6.12.1	Detailed Description	38
6.12.2	Field Documentation	38
6.12.2.1	cond	38
6.12.2.2	mutex	38
6.12.2.3	new_value_ready	38
6.13	lspg_starttransfer_struct Struct Reference	38
6.13.1	Detailed Description	39
6.13.2	Field Documentation	39
6.13.2.1	cond	39
6.13.2.2	mutex	39
6.13.2.3	new_value_ready	39
6.13.2.4	no_rows_returned	39
6.13.2.5	starttransfer	39
6.14	lspg_wait_for_detector_struct Struct Reference	39
6.14.1	Detailed Description	40
6.14.2	Field Documentation	40
6.14.2.1	cond	40
6.14.2.2	mutex	40
6.14.2.3	new_value_ready	40
6.15	lspg_waitcryo_struct Struct Reference	40
6.15.1	Detailed Description	40
6.15.2	Field Documentation	40
6.15.2.1	cond	40



6.15.2.2	mutex	41
6.15.2.3	new_value_ready	41
6.16	IspgQueryQueueStruct Struct Reference	41
6.16.1	Detailed Description	41
6.16.2	Field Documentation	41
6.16.2.1	onResponse	41
6.16.2.2	qs	41
6.17	Ispmac_ascii_buffers_struct Struct Reference	42
6.17.1	Detailed Description	42
6.17.2	Field Documentation	42
6.17.2.1	command_buf	42
6.17.2.2	command_buf_cc	42
6.17.2.3	command_str	42
6.17.2.4	response_buf	42
6.17.2.5	response_n	42
6.17.2.6	response_str	42
6.18	Ispmac_bi_struct Struct Reference	42
6.18.1	Detailed Description	43
6.18.2	Field Documentation	43
6.18.2.1	changeEventOff	43
6.18.2.2	changeEventOn	43
6.18.2.3	first_time	43
6.18.2.4	mask	43
6.18.2.5	mutex	44
6.18.2.6	position	44
6.18.2.7	previous	44
6.18.2.8	ptr	44
6.19	Ispmac_cmd_queue_struct Struct Reference	44
6.19.1	Detailed Description	44
6.19.2	Field Documentation	45
6.19.2.1	event	45
6.19.2.2	no_reply	45
6.19.2.3	onResponse	45
6.19.2.4	pcmd	45
6.19.2.5	time_sent	45
6.20	Ispmac_dpascii_queue_struct Struct Reference	45
6.20.1	Detailed Description	45
6.20.2	Field Documentation	45
6.20.2.1	event	45
6.20.2.2	pl	46

6.21	<a href="#">lspmac_motor_struct Struct Reference</a>	46
6.21.1	<a href="#">Detailed Description</a>	48
6.21.2	<a href="#">Field Documentation</a>	48
6.21.2.1	<a href="#">active</a>	48
6.21.2.2	<a href="#">active_init</a>	48
6.21.2.3	<a href="#">actual_pos_cnts</a>	48
6.21.2.4	<a href="#">actual_pos_cnts_p</a>	48
6.21.2.5	<a href="#">axis</a>	48
6.21.2.6	<a href="#">command_sent</a>	48
6.21.2.7	<a href="#">cond</a>	48
6.21.2.8	<a href="#">coord_num</a>	48
6.21.2.9	<a href="#">dac_mvar</a>	49
6.21.2.10	<a href="#">home</a>	49
6.21.2.11	<a href="#">homing</a>	49
6.21.2.12	<a href="#">inactive_init</a>	49
6.21.2.13	<a href="#">lut</a>	49
6.21.2.14	<a href="#">max_accel</a>	49
6.21.2.15	<a href="#">max_speed</a>	49
6.21.2.16	<a href="#">motion_seen</a>	49
6.21.2.17	<a href="#">motor_num</a>	49
6.21.2.18	<a href="#">moveAbs</a>	50
6.21.2.19	<a href="#">mutex</a>	50
6.21.2.20	<a href="#">name</a>	50
6.21.2.21	<a href="#">nlut</a>	50
6.21.2.22	<a href="#">not_done</a>	50
6.21.2.23	<a href="#">position</a>	50
6.21.2.24	<a href="#">pq</a>	50
6.21.2.25	<a href="#">printf_fmt</a>	50
6.21.2.26	<a href="#">read</a>	50
6.21.2.27	<a href="#">read_mask</a>	51
6.21.2.28	<a href="#">read_ptr</a>	51
6.21.2.29	<a href="#">redis_fmt</a>	51
6.21.2.30	<a href="#">redis_position</a>	51
6.21.2.31	<a href="#">reported_position</a>	51
6.21.2.32	<a href="#">requested_pos_cnts</a>	51
6.21.2.33	<a href="#">requested_position</a>	51
6.21.2.34	<a href="#">status1</a>	51
6.21.2.35	<a href="#">status1_p</a>	51
6.21.2.36	<a href="#">status2</a>	52
6.21.2.37	<a href="#">status2_p</a>	52

6.21.2.38	status_str	52
6.21.2.39	u2c	52
6.21.2.40	unit	52
6.21.2.41	update_resolution	52
6.21.2.42	win	52
6.21.2.43	write_fmt	52
6.22	Isredis_obj_struct Struct Reference	52
6.22.1	Detailed Description	53
6.22.2	Field Documentation	53
6.22.2.1	avalue	53
6.22.2.2	bvalue	53
6.22.2.3	cond	54
6.22.2.4	cvalue	54
6.22.2.5	dvalue	54
6.22.2.6	events_name	54
6.22.2.7	hits	54
6.22.2.8	key	54
6.22.2.9	lvalue	54
6.22.2.10	mutex	54
6.22.2.11	next	54
6.22.2.12	valid	55
6.22.2.13	value	55
6.22.2.14	value_length	55
6.22.2.15	wait_for_me	55
6.23	Istimer_list_struct Struct Reference	55
6.23.1	Detailed Description	56
6.23.2	Field Documentation	56
6.23.2.1	delay_nsecs	56
6.23.2.2	delay_secs	56
6.23.2.3	event	56
6.23.2.4	init_nsecs	56
6.23.2.5	init_secs	56
6.23.2.6	last_nsecs	56
6.23.2.7	last_secs	56
6.23.2.8	ncalls	57
6.23.2.9	next_nsecs	57
6.23.2.10	next_secs	57
6.23.2.11	shots	57
6.24	md2StatusStruct Struct Reference	57
6.24.1	Detailed Description	58

6.24.2	Field Documentation	59
6.24.2.1	acc11c_1	59
6.24.2.2	acc11c_2	59
6.24.2.3	acc11c_3	59
6.24.2.4	acc11c_5	59
6.24.2.5	acc11c_6	59
6.24.2.6	alignx_act_pos	59
6.24.2.7	alignx_status_1	59
6.24.2.8	alignx_status_2	59
6.24.2.9	aligny_act_pos	59
6.24.2.10	aligny_status_1	59
6.24.2.11	aligny_status_2	59
6.24.2.12	alignz_act_pos	59
6.24.2.13	alignz_status_1	60
6.24.2.14	alignz_status_2	60
6.24.2.15	analyzer_act_pos	60
6.24.2.16	analyzer_status_1	60
6.24.2.17	analyzer_status_2	60
6.24.2.18	aperturey_act_pos	60
6.24.2.19	aperturey_status_1	60
6.24.2.20	aperturey_status_2	60
6.24.2.21	aperturez_act_pos	60
6.24.2.22	aperturez_status_1	60
6.24.2.23	aperturez_status_2	60
6.24.2.24	back_dac	60
6.24.2.25	copy_act_pos	61
6.24.2.26	copy_status_1	61
6.24.2.27	copy_status_2	61
6.24.2.28	capz_act_pos	61
6.24.2.29	capz_status_1	61
6.24.2.30	capz_status_2	61
6.24.2.31	centerx_act_pos	61
6.24.2.32	centerx_status_1	61
6.24.2.33	centerx_status_2	61
6.24.2.34	centery_act_pos	61
6.24.2.35	centery_status_1	61
6.24.2.36	centery_status_2	61
6.24.2.37	dummy1	62
6.24.2.38	dummy2	62
6.24.2.39	dummy3	62

6.24.2.40 dummy4 . . . . .	62
6.24.2.41 dummy5 . . . . .	62
6.24.2.42 dummy6 . . . . .	62
6.24.2.43 dummy7 . . . . .	62
6.24.2.44 dummy8 . . . . .	62
6.24.2.45 dummy9 . . . . .	62
6.24.2.46 dummyA . . . . .	62
6.24.2.47 dummyB . . . . .	62
6.24.2.48 front_dac . . . . .	62
6.24.2.49 fs_has_opened . . . . .	63
6.24.2.50 fs_has_opened_globally . . . . .	63
6.24.2.51 fs_is_open . . . . .	63
6.24.2.52 kappa_act_pos . . . . .	63
6.24.2.53 kappa_status_1 . . . . .	63
6.24.2.54 kappa_status_2 . . . . .	63
6.24.2.55 moving_flags . . . . .	63
6.24.2.56 number_passes . . . . .	63
6.24.2.57 omega_act_pos . . . . .	63
6.24.2.58 omega_status_1 . . . . .	63
6.24.2.59 omega_status_2 . . . . .	63
6.24.2.60 phi_act_pos . . . . .	63
6.24.2.61 phi_status_1 . . . . .	64
6.24.2.62 phi_status_2 . . . . .	64
6.24.2.63 phiscan . . . . .	64
6.24.2.64 scint_act_pos . . . . .	64
6.24.2.65 scint_piezo . . . . .	64
6.24.2.66 scint_status_1 . . . . .	64
6.24.2.67 scint_status_2 . . . . .	64
6.24.2.68 zoom_act_pos . . . . .	64
6.24.2.69 zoom_status_1 . . . . .	64
6.24.2.70 zoom_status_2 . . . . .	64
6.25 tagEthernetCmd Struct Reference . . . . .	64
6.25.1 Detailed Description . . . . .	65
6.25.2 Field Documentation . . . . .	65
6.25.2.1 bData . . . . .	65
6.25.2.2 Request . . . . .	65
6.25.2.3 RequestType . . . . .	65
6.25.2.4 wIndex . . . . .	65
6.25.2.5 wLength . . . . .	65
6.25.2.6 wValue . . . . .	66

<b>7</b>	<b>File Documentation</b>	<b>67</b>
7.1	iniParser.py File Reference	67
7.2	kvredis.c File Reference	67
7.2.1	Macro Definition Documentation	69
7.2.1.1	LS_PG_QUERY_QUEUE_LENGTH	69
7.2.1.2	LS_PG_QUERY_STRING_LENGTH	69
7.2.1.3	LS_PG_STATE_IDLE	69
7.2.1.4	LS_PG_STATE_INIT	69
7.2.1.5	LS_PG_STATE_INIT_POLL	70
7.2.1.6	LS_PG_STATE_RECV	70
7.2.1.7	LS_PG_STATE_RESET	70
7.2.1.8	LS_PG_STATE_RESET_POLL	70
7.2.1.9	LS_PG_STATE_SEND	70
7.2.1.10	LS_PG_STATE_SEND_FLUSH	70
7.2.2	Typedef Documentation	70
7.2.2.1	lspg_query_queue_t	70
7.2.3	Function Documentation	70
7.2.3.1	addRead	70
7.2.3.2	addWrite	70
7.2.3.3	cleanup	71
7.2.3.4	debugCB	71
7.2.3.5	delRead	71
7.2.3.6	delWrite	72
7.2.3.7	fd_service	72
7.2.3.8	lspg_allkvs_cb	72
7.2.3.9	lspg_flush	73
7.2.3.10	lspg_next_state	73
7.2.3.11	lspg_notice_processor	74
7.2.3.12	lspg_pg_connect	74
7.2.3.13	lspg_pg_service	75
7.2.3.14	lspg_query_next	77
7.2.3.15	lspg_query_push	77
7.2.3.16	lspg_query_reply_next	77
7.2.3.17	lspg_query_reply_peek	78
7.2.3.18	lspg_receive	78
7.2.3.19	lspg_send_next_query	79
7.2.3.20	main	79
7.2.3.21	redisDisconnectCB	81
7.2.4	Variable Documentation	81
7.2.4.1	cmdac	81

7.2.4.2	<a href="#">cmdfd</a>	81
7.2.4.3	<a href="#">kvseq</a>	81
7.2.4.4	<a href="#">ls_pg_state</a>	81
7.2.4.5	<a href="#">lspg_connectPoll_response</a>	82
7.2.4.6	<a href="#">lspg_query_queue</a>	82
7.2.4.7	<a href="#">lspg_query_queue_off</a>	82
7.2.4.8	<a href="#">lspg_query_queue_on</a>	82
7.2.4.9	<a href="#">lspg_query_queue_reply</a>	82
7.2.4.10	<a href="#">lspg_resetPoll_response</a>	82
7.2.4.11	<a href="#">lspgfd</a>	82
7.2.4.12	<a href="#">now</a>	82
7.2.4.13	<a href="#">q</a>	83
7.2.4.14	<a href="#">subac</a>	83
7.2.4.15	<a href="#">subfd</a>	83
7.3	<a href="#">lsevents.c File Reference</a>	83
7.3.1	<a href="#">Detailed Description</a>	84
7.3.2	<a href="#">Macro Definition Documentation</a>	84
7.3.2.1	<a href="#">LSEVENTS_QUEUE_LENGTH</a>	84
7.3.3	<a href="#">Typedef Documentation</a>	85
7.3.3.1	<a href="#">lsevents_listener_t</a>	85
7.3.3.2	<a href="#">lsevents_queue_t</a>	85
7.3.4	<a href="#">Function Documentation</a>	85
7.3.4.1	<a href="#">lsevents_add_listener</a>	85
7.3.4.2	<a href="#">lsevents_init</a>	86
7.3.4.3	<a href="#">lsevents_remove_listener</a>	86
7.3.4.4	<a href="#">lsevents_run</a>	86
7.3.4.5	<a href="#">lsevents_send_event</a>	87
7.3.4.6	<a href="#">lsevents_worker</a>	87
7.3.5	<a href="#">Variable Documentation</a>	88
7.3.5.1	<a href="#">lsevents_listener_mutex</a>	88
7.3.5.2	<a href="#">lsevents_listeners_p</a>	88
7.3.5.3	<a href="#">lsevents_queue</a>	88
7.3.5.4	<a href="#">lsevents_queue_cond</a>	88
7.3.5.5	<a href="#">lsevents_queue_mutex</a>	88
7.3.5.6	<a href="#">lsevents_queue_off</a>	89
7.3.5.7	<a href="#">lsevents_queue_on</a>	89
7.3.5.8	<a href="#">lsevents_thread</a>	89
7.4	<a href="#">islogging.c File Reference</a>	89
7.4.1	<a href="#">Detailed Description</a>	90
7.4.2	<a href="#">Macro Definition Documentation</a>	90

7.4.2.1	LSLOGGING_FILE_NAME	90
7.4.2.2	LSLOGGING_MSG_LENGTH	90
7.4.2.3	LSLOGGING_QUEUE_LENGTH	91
7.4.3	Typedef Documentation	91
7.4.3.1	lslogging_queue_t	91
7.4.4	Function Documentation	91
7.4.4.1	lslogging_init	91
7.4.4.2	lslogging_log_message	91
7.4.4.3	lslogging_run	92
7.4.4.4	lslogging_worker	92
7.4.5	Variable Documentation	92
7.4.5.1	lslogging_cond	92
7.4.5.2	lslogging_file	92
7.4.5.3	lslogging_mutex	93
7.4.5.4	lslogging_off	93
7.4.5.5	lslogging_on	93
7.4.5.6	lslogging_queue	93
7.4.5.7	lslogging_thread	93
7.5	lspg.c File Reference	93
7.5.1	Detailed Description	98
7.5.2	Macro Definition Documentation	98
7.5.2.1	LS_PG_QUERY_QUEUE_LENGTH	98
7.5.2.2	LS_PG_STATE_IDLE	99
7.5.2.3	LS_PG_STATE_INIT	99
7.5.2.4	LS_PG_STATE_INIT_POLL	99
7.5.2.5	LS_PG_STATE_RECV	99
7.5.2.6	LS_PG_STATE_RESET	99
7.5.2.7	LS_PG_STATE_RESET_POLL	99
7.5.2.8	LS_PG_STATE_SEND	99
7.5.2.9	LS_PG_STATE_SEND_FLUSH	99
7.5.3	Typedef Documentation	99
7.5.3.1	lspg_lock_detector_t	99
7.5.3.2	lspg_lock_diffractionmeter_t	99
7.5.3.3	lspg_seq_run_prep_t	99
7.5.3.4	lspg_wait_for_detector_t	99
7.5.4	Function Documentation	100
7.5.4.1	lspg_array2ptrs	100
7.5.4.2	lspg_cmd_cb	101
7.5.4.3	lspg_demandairrights_all	102
7.5.4.4	lspg_demandairrights_call	102



7.5.4.5	<a href="#">lspg_demandairrights_cb</a>	102
7.5.4.6	<a href="#">lspg_demandairrights_init</a>	102
7.5.4.7	<a href="#">lspg_demandairrights_wait</a>	102
7.5.4.8	<a href="#">lspg_flush</a>	103
7.5.4.9	<a href="#">lspg_getcenter_all</a>	103
7.5.4.10	<a href="#">lspg_getcenter_call</a>	103
7.5.4.11	<a href="#">lspg_getcenter_cb</a>	103
7.5.4.12	<a href="#">lspg_getcenter_done</a>	104
7.5.4.13	<a href="#">lspg_getcenter_init</a>	104
7.5.4.14	<a href="#">lspg_getcenter_wait</a>	105
7.5.4.15	<a href="#">lspg_getcurrentsampleid_call</a>	105
7.5.4.16	<a href="#">lspg_getcurrentsampleid_cb</a>	105
7.5.4.17	<a href="#">lspg_getcurrentsampleid_init</a>	106
7.5.4.18	<a href="#">lspg_getcurrentsampleid_read</a>	106
7.5.4.19	<a href="#">lspg_getcurrentsampleid_wait_for_id</a>	106
7.5.4.20	<a href="#">lspg_init</a>	106
7.5.4.21	<a href="#">lspg_lock_detector_all</a>	107
7.5.4.22	<a href="#">lspg_lock_detector_call</a>	107
7.5.4.23	<a href="#">lspg_lock_detector_cb</a>	107
7.5.4.24	<a href="#">lspg_lock_detector_done</a>	107
7.5.4.25	<a href="#">lspg_lock_detector_init</a>	108
7.5.4.26	<a href="#">lspg_lock_detector_wait</a>	108
7.5.4.27	<a href="#">lspg_lock_diffractionmeter_all</a>	108
7.5.4.28	<a href="#">lspg_lock_diffractionmeter_call</a>	108
7.5.4.29	<a href="#">lspg_lock_diffractionmeter_cb</a>	108
7.5.4.30	<a href="#">lspg_lock_diffractionmeter_done</a>	109
7.5.4.31	<a href="#">lspg_lock_diffractionmeter_init</a>	109
7.5.4.32	<a href="#">lspg_lock_diffractionmeter_wait</a>	109
7.5.4.33	<a href="#">lspg_next_state</a>	109
7.5.4.34	<a href="#">lspg_nextaction_cb</a>	110
7.5.4.35	<a href="#">lspg_nextsample_all</a>	111
7.5.4.36	<a href="#">lspg_nextsample_call</a>	111
7.5.4.37	<a href="#">lspg_nextsample_cb</a>	111
7.5.4.38	<a href="#">lspg_nextsample_done</a>	112
7.5.4.39	<a href="#">lspg_nextsample_init</a>	112
7.5.4.40	<a href="#">lspg_nextsample_wait</a>	112
7.5.4.41	<a href="#">lspg_nextshot_call</a>	112
7.5.4.42	<a href="#">lspg_nextshot_cb</a>	113
7.5.4.43	<a href="#">lspg_nextshot_done</a>	116
7.5.4.44	<a href="#">lspg_nextshot_init</a>	117

7.5.4.45	<a href="#">lspg_nextshot_wait</a>	117
7.5.4.46	<a href="#">lspg_notice_processor</a>	117
7.5.4.47	<a href="#">lspg_pg_connect</a>	117
7.5.4.48	<a href="#">lspg_pg_service</a>	118
7.5.4.49	<a href="#">lspg_query_next</a>	120
7.5.4.50	<a href="#">lspg_query_push</a>	120
7.5.4.51	<a href="#">lspg_query_reply_next</a>	121
7.5.4.52	<a href="#">lspg_query_reply_peek</a>	121
7.5.4.53	<a href="#">lspg_receive</a>	121
7.5.4.54	<a href="#">lspg_run</a>	122
7.5.4.55	<a href="#">lspg_send_next_query</a>	122
7.5.4.56	<a href="#">lspg_seq_run_prep_all</a>	123
7.5.4.57	<a href="#">lspg_seq_run_prep_call</a>	124
7.5.4.58	<a href="#">lspg_seq_run_prep_cb</a>	124
7.5.4.59	<a href="#">lspg_seq_run_prep_done</a>	124
7.5.4.60	<a href="#">lspg_seq_run_prep_init</a>	124
7.5.4.61	<a href="#">lspg_seq_run_prep_wait</a>	125
7.5.4.62	<a href="#">lspg_sig_service</a>	125
7.5.4.63	<a href="#">lspg_starttransfer_all</a>	125
7.5.4.64	<a href="#">lspg_starttransfer_call</a>	126
7.5.4.65	<a href="#">lspg_starttransfer_cb</a>	126
7.5.4.66	<a href="#">lspg_starttransfer_done</a>	126
7.5.4.67	<a href="#">lspg_starttransfer_init</a>	126
7.5.4.68	<a href="#">lspg_starttransfer_wait</a>	127
7.5.4.69	<a href="#">lspg_wait_for_detector_all</a>	127
7.5.4.70	<a href="#">lspg_wait_for_detector_call</a>	127
7.5.4.71	<a href="#">lspg_wait_for_detector_cb</a>	127
7.5.4.72	<a href="#">lspg_wait_for_detector_done</a>	127
7.5.4.73	<a href="#">lspg_wait_for_detector_init</a>	128
7.5.4.74	<a href="#">lspg_wait_for_detector_wait</a>	128
7.5.4.75	<a href="#">lspg_waitcryo_all</a>	128
7.5.4.76	<a href="#">lspg_waitcryo_cb</a>	128
7.5.4.77	<a href="#">lspg_waitcryo_init</a>	129
7.5.4.78	<a href="#">lspg_worker</a>	129
7.5.4.79	<a href="#">lspmac_sample_detector_cb</a>	130
7.5.5	<a href="#">Variable Documentation</a>	130
7.5.5.1	<a href="#">ls_pg_state</a>	130
7.5.5.2	<a href="#">lspg_connectPoll_response</a>	130
7.5.5.3	<a href="#">lspg_demandairrights</a>	130
7.5.5.4	<a href="#">lspg_getcenter</a>	130

7.5.5.5	<a href="#">lspg_getcurrentsampleid</a>	130
7.5.5.6	<a href="#">lspg_lock_detector</a>	131
7.5.5.7	<a href="#">lspg_lock_diffractionmeter</a>	131
7.5.5.8	<a href="#">lspg_nextsample</a>	131
7.5.5.9	<a href="#">lspg_nextshot</a>	131
7.5.5.10	<a href="#">lspg_query_queue</a>	131
7.5.5.11	<a href="#">lspg_query_queue_off</a>	131
7.5.5.12	<a href="#">lspg_query_queue_on</a>	131
7.5.5.13	<a href="#">lspg_query_queue_reply</a>	131
7.5.5.14	<a href="#">lspg_queue_cond</a>	131
7.5.5.15	<a href="#">lspg_queue_mutex</a>	132
7.5.5.16	<a href="#">lspg_resetPoll_response</a>	132
7.5.5.17	<a href="#">lspg_seq_run_prep</a>	132
7.5.5.18	<a href="#">lspg_starttransfer</a>	132
7.5.5.19	<a href="#">lspg_thread</a>	132
7.5.5.20	<a href="#">lspg_wait_for_detector</a>	132
7.5.5.21	<a href="#">lspg_waitcryo</a>	132
7.5.5.22	<a href="#">lspgfd</a>	132
7.5.5.23	<a href="#">now</a>	132
7.5.5.24	<a href="#">q</a>	133
7.6	<a href="#">lspmac.c File Reference</a>	133
7.6.1	<a href="#">Detailed Description</a>	141
7.6.2	<a href="#">Macro Definition Documentation</a>	141
7.6.2.1	<a href="#">LS_PMAC_STATE_CR</a>	141
7.6.2.2	<a href="#">LS_PMAC_STATE_DETACHED</a>	141
7.6.2.3	<a href="#">LS_PMAC_STATE_GB</a>	141
7.6.2.4	<a href="#">LS_PMAC_STATE_GMR</a>	142
7.6.2.5	<a href="#">LS_PMAC_STATE_IDLE</a>	142
7.6.2.6	<a href="#">LS_PMAC_STATE_RESET</a>	142
7.6.2.7	<a href="#">LS_PMAC_STATE_RR</a>	142
7.6.2.8	<a href="#">LS_PMAC_STATE_SC</a>	142
7.6.2.9	<a href="#">LS_PMAC_STATE_WACK</a>	142
7.6.2.10	<a href="#">LS_PMAC_STATE_WACK_CC</a>	142
7.6.2.11	<a href="#">LS_PMAC_STATE_WACK_NFR</a>	142
7.6.2.12	<a href="#">LS_PMAC_STATE_WACK_RR</a>	142
7.6.2.13	<a href="#">LS_PMAC_STATE_WCR</a>	142
7.6.2.14	<a href="#">LS_PMAC_STATE_WGB</a>	142
7.6.2.15	<a href="#">LSPMAC_DPASCII_QUEUE_LENGTH</a>	142
7.6.2.16	<a href="#">LSPMAC_PRESET_REGEX</a>	143
7.6.2.17	<a href="#">PMAC_CMD_QUEUE_LENGTH</a>	143

7.6.2.18	<a href="#">pmac_cmd_size</a>	143
7.6.2.19	<a href="#">PMAC_MIN_CMD_TIME</a>	143
7.6.2.20	<a href="#">PMACPORT</a>	143
7.6.2.21	<a href="#">VR_CTRL_RESPONSE</a>	143
7.6.2.22	<a href="#">VR_DOWNLOAD</a>	143
7.6.2.23	<a href="#">VR_FWDOWNLOAD</a>	143
7.6.2.24	<a href="#">VR_IPADDRESS</a>	143
7.6.2.25	<a href="#">VR_PMAC_FLUSH</a>	143
7.6.2.26	<a href="#">VR_PMAC_GETBUFFER</a>	143
7.6.2.27	<a href="#">VR_PMAC_GETLINE</a>	144
7.6.2.28	<a href="#">VR_PMAC_GETMEM</a>	144
7.6.2.29	<a href="#">VR_PMAC_GETRESPONSE</a>	144
7.6.2.30	<a href="#">VR_PMAC_PORT</a>	144
7.6.2.31	<a href="#">VR_PMAC_READREADY</a>	144
7.6.2.32	<a href="#">VR_PMAC_SENDCTRLCHAR</a>	144
7.6.2.33	<a href="#">VR_PMAC_SENDLINE</a>	144
7.6.2.34	<a href="#">VR_PMAC_SETBIT</a>	144
7.6.2.35	<a href="#">VR_PMAC_SETBITS</a>	144
7.6.2.36	<a href="#">VR_PMAC_SETMEM</a>	144
7.6.2.37	<a href="#">VR_PMAC_WRITEBUFFER</a>	144
7.6.2.38	<a href="#">VR_PMAC_WRITEERROR</a>	144
7.6.2.39	<a href="#">VR_UPLOAD</a>	145
7.6.3	<a href="#">Typedef Documentation</a>	145
7.6.3.1	<a href="#">lspmac_ascii_buffers_t</a>	145
7.6.3.2	<a href="#">lspmac_dpascii_queue_t</a>	145
7.6.3.3	<a href="#">md2_status_t</a>	145
7.6.4	<a href="#">Function Documentation</a>	145
7.6.4.1	<a href="#">_lspmac_motor_init</a>	145
7.6.4.2	<a href="#">cleanstr</a>	146
7.6.4.3	<a href="#">hex_dump</a>	146
7.6.4.4	<a href="#">lsConnect</a>	146
7.6.4.5	<a href="#">lspmac_asciicmdCB</a>	147
7.6.4.6	<a href="#">lspmac_backLight_down_cb</a>	147
7.6.4.7	<a href="#">lspmac_backLight_up_cb</a>	148
7.6.4.8	<a href="#">lspmac_bi_init</a>	148
7.6.4.9	<a href="#">lspmac_blight_lut_setup</a>	148
7.6.4.10	<a href="#">lspmac_bo_init</a>	149
7.6.4.11	<a href="#">lspmac_bo_read</a>	149
7.6.4.12	<a href="#">lspmac_command_done_cb</a>	150
7.6.4.13	<a href="#">lspmac_cryoSwitchChanged_cb</a>	150

7.6.4.14	<a href="#">lspmac_dac_init</a>	150
7.6.4.15	<a href="#">lspmac_dac_read</a>	151
7.6.4.16	<a href="#">lspmac_Error</a>	151
7.6.4.17	<a href="#">lspmac_flight_lut_setup</a>	152
7.6.4.18	<a href="#">lspmac_fscint_lut_setup</a>	152
7.6.4.19	<a href="#">lspmac_fshut_init</a>	153
7.6.4.20	<a href="#">lspmac_get_ascii</a>	153
7.6.4.21	<a href="#">lspmac_get_ascii_cb</a>	153
7.6.4.22	<a href="#">lspmac_get_status</a>	155
7.6.4.23	<a href="#">lspmac_get_status_cb</a>	155
7.6.4.24	<a href="#">lspmac_GetAllIVars</a>	158
7.6.4.25	<a href="#">lspmac_GetAllIVarsCB</a>	158
7.6.4.26	<a href="#">lspmac_GetAllMVars</a>	158
7.6.4.27	<a href="#">lspmac_GetAllMVarsCB</a>	159
7.6.4.28	<a href="#">lspmac_getBIPosition</a>	159
7.6.4.29	<a href="#">lspmac_Getmem</a>	159
7.6.4.30	<a href="#">lspmac_GetmemReplyCB</a>	159
7.6.4.31	<a href="#">lspmac_getPosition</a>	160
7.6.4.32	<a href="#">lspmac_GetShortReplyCB</a>	160
7.6.4.33	<a href="#">lspmac_home1_queue</a>	161
7.6.4.34	<a href="#">lspmac_home2_queue</a>	162
7.6.4.35	<a href="#">lspmac_init</a>	162
7.6.4.36	<a href="#">lspmac_jogabs_queue</a>	165
7.6.4.37	<a href="#">lspmac_light_zoom_cb</a>	166
7.6.4.38	<a href="#">lspmac_lut</a>	166
7.6.4.39	<a href="#">lspmac_more_ascii_cb</a>	167
7.6.4.40	<a href="#">lspmac_motor_init</a>	167
7.6.4.41	<a href="#">lspmac_move_or_jog_abs_queue</a>	168
7.6.4.42	<a href="#">lspmac_move_or_jog_preset_queue</a>	170
7.6.4.43	<a href="#">lspmac_move_preset_queue</a>	170
7.6.4.44	<a href="#">lspmac_moveabs_blight_factor_queue</a>	170
7.6.4.45	<a href="#">lspmac_moveabs_bo_queue</a>	171
7.6.4.46	<a href="#">lspmac_moveabs_flight_factor_queue</a>	171
7.6.4.47	<a href="#">lspmac_moveabs_frontlight_oo_queue</a>	172
7.6.4.48	<a href="#">lspmac_moveabs_fshut_queue</a>	172
7.6.4.49	<a href="#">lspmac_moveabs_queue</a>	172
7.6.4.50	<a href="#">lspmac_moveabs_timed_queue</a>	173
7.6.4.51	<a href="#">lspmac_moveabs_wait</a>	174
7.6.4.52	<a href="#">lspmac_movedac_queue</a>	174
7.6.4.53	<a href="#">lspmac_movezoom_queue</a>	175

7.6.4.54	<a href="#">lspmac_next_state</a>	175
7.6.4.55	<a href="#">lspmac_pmacmotor_read</a>	177
7.6.4.56	<a href="#">lspmac_pop_queue</a>	180
7.6.4.57	<a href="#">lspmac_pop_reply</a>	180
7.6.4.58	<a href="#">lspmac_push_queue</a>	181
7.6.4.59	<a href="#">lspmac_Reset</a>	181
7.6.4.60	<a href="#">lspmac_reset_queue</a>	181
7.6.4.61	<a href="#">lspmac_rlut</a>	182
7.6.4.62	<a href="#">lspmac_run</a>	182
7.6.4.63	<a href="#">lspmac_scint_dried_cb</a>	184
7.6.4.64	<a href="#">lspmac_scint_inPosition_cb</a>	184
7.6.4.65	<a href="#">lspmac_send_command</a>	185
7.6.4.66	<a href="#">lspmac_sendcmd</a>	185
7.6.4.67	<a href="#">lspmac_sendcmd_nocb</a>	186
7.6.4.68	<a href="#">lspmac_SendControlReplyPrintCB</a>	186
7.6.4.69	<a href="#">lspmac_Service</a>	187
7.6.4.70	<a href="#">lspmac_shutter_read</a>	189
7.6.4.71	<a href="#">lspmac_SockFlush</a>	190
7.6.4.72	<a href="#">lspmac_SockGetmem</a>	190
7.6.4.73	<a href="#">lspmac_SockSendControlCharPrint</a>	190
7.6.4.74	<a href="#">lspmac_SockSendDPline</a>	190
7.6.4.75	<a href="#">lspmac_SockSendDPqueue</a>	191
7.6.4.76	<a href="#">lspmac_SockSendline</a>	191
7.6.4.77	<a href="#">lspmac_SockSendline_nr</a>	192
7.6.4.78	<a href="#">lspmac_soft_motor_init</a>	192
7.6.4.79	<a href="#">lspmac_soft_motor_read</a>	192
7.6.4.80	<a href="#">lspmac_test_preset</a>	192
7.6.4.81	<a href="#">lspmac_video_rotate</a>	193
7.6.4.82	<a href="#">lspmac_worker</a>	193
7.6.4.83	<a href="#">lspmac_zoom_lut_setup</a>	194
7.6.5	<a href="#">Variable Documentation</a>	194
7.6.5.1	<a href="#">alignx</a>	194
7.6.5.2	<a href="#">aligny</a>	195
7.6.5.3	<a href="#">alignz</a>	195
7.6.5.4	<a href="#">anal</a>	195
7.6.5.5	<a href="#">apery</a>	195
7.6.5.6	<a href="#">aperz</a>	195
7.6.5.7	<a href="#">arm_parked</a>	195
7.6.5.8	<a href="#">blight</a>	195
7.6.5.9	<a href="#">blight_down</a>	195

7.6.5.10	blight_f	195
7.6.5.11	blight_ud	196
7.6.5.12	blight_up	196
7.6.5.13	capy	196
7.6.5.14	capz	196
7.6.5.15	cenx	196
7.6.5.16	ceny	196
7.6.5.17	cr_cmd	196
7.6.5.18	cryo	196
7.6.5.19	cryo_back	196
7.6.5.20	cryo_switch	197
7.6.5.21	dbmem	197
7.6.5.22	dbmemIn	197
7.6.5.23	dryer	197
7.6.5.24	etel_init_ok	197
7.6.5.25	etel_on	197
7.6.5.26	etel_ready	197
7.6.5.27	ethCmdOff	197
7.6.5.28	ethCmdOn	197
7.6.5.29	ethCmdQueue	198
7.6.5.30	ethCmdReply	198
7.6.5.31	flight	198
7.6.5.32	flight_f	198
7.6.5.33	flight_oo	198
7.6.5.34	fluo	198
7.6.5.35	fluor_back	198
7.6.5.36	fscint	198
7.6.5.37	fshut	198
7.6.5.38	gb_cmd	199
7.6.5.39	getivars	199
7.6.5.40	getmvars	199
7.6.5.41	hp_air	199
7.6.5.42	kappa	199
7.6.5.43	lp_air	199
7.6.5.44	ls_pmac_state	199
7.6.5.45	lspmac_ascii_buffers	199
7.6.5.46	lspmac_ascii_buffers_mutex	199
7.6.5.47	lspmac_ascii_busy	199
7.6.5.48	lspmac_ascii_mutex	200
7.6.5.49	lspmac_bis	200

7.6.5.50	<code>lspmac_dpascii_off</code>	200
7.6.5.51	<code>lspmac_dpascii_on</code>	200
7.6.5.52	<code>lspmac_dpascii_queue</code>	200
7.6.5.53	<code>lspmac_md2_init</code>	200
7.6.5.54	<code>lspmac_motors</code>	200
7.6.5.55	<code>lspmac_moving_cond</code>	200
7.6.5.56	<code>lspmac_moving_flags</code>	200
7.6.5.57	<code>lspmac_moving_mutex</code>	200
7.6.5.58	<code>lspmac_nbis</code>	201
7.6.5.59	<code>lspmac_nmotors</code>	201
7.6.5.60	<code>lspmac_shutter_cond</code>	201
7.6.5.61	<code>lspmac_shutter_has_opened</code>	201
7.6.5.62	<code>lspmac_shutter_mutex</code>	201
7.6.5.63	<code>lspmac_shutter_state</code>	201
7.6.5.64	<code>lspmac_status_last_time</code>	201
7.6.5.65	<code>lspmac_status_time</code>	201
7.6.5.66	<code>md2_status</code>	201
7.6.5.67	<code>md2_status_mutex</code>	202
7.6.5.68	<code>minikappa_ok</code>	202
7.6.5.69	<code>now</code>	202
7.6.5.70	<code>omega</code>	202
7.6.5.71	<code>omega_zero_search</code>	202
7.6.5.72	<code>omega_zero_time</code>	202
7.6.5.73	<code>omega_zero_velocity</code>	202
7.6.5.74	<code>phi</code>	202
7.6.5.75	<code>pmac_error_strs</code>	202
7.6.5.76	<code>pmac_queue_cond</code>	203
7.6.5.77	<code>pmac_queue_mutex</code>	203
7.6.5.78	<code>pmac_thread</code>	203
7.6.5.79	<code>pmacfd</code>	203
7.6.5.80	<code>rr_cmd</code>	203
7.6.5.81	<code>sample_detected</code>	203
7.6.5.82	<code>scint</code>	203
7.6.5.83	<code>shutter_open</code>	204
7.6.5.84	<code>smart_mag_err</code>	204
7.6.5.85	<code>smart_mag_off</code>	204
7.6.5.86	<code>smart_mag_on</code>	204
7.6.5.87	<code>smart_mag_oo</code>	204
7.6.5.88	<code>zoom</code>	204
7.7	Isredis.c File Reference	204



7.7.1	Detailed Description	206
7.7.2	Function Documentation	207
7.7.2.1	_lsredis_get_obj	207
7.7.2.2	_lsredis_set_value	208
7.7.2.3	lsredis_addRead	209
7.7.2.4	lsredis_addWrite	209
7.7.2.5	lsredis_cleanup	209
7.7.2.6	lsredis_cmpnstr	210
7.7.2.7	lsredis_cmpstr	210
7.7.2.8	lsredis_debugCB	210
7.7.2.9	lsredis_delRead	211
7.7.2.10	lsredis_delWrite	211
7.7.2.11	lsredis_fd_service	211
7.7.2.12	lsredis_find_preset	212
7.7.2.13	lsredis_get_obj	212
7.7.2.14	lsredis_get_string_array	213
7.7.2.15	lsredis_getb	213
7.7.2.16	lsredis_getc	213
7.7.2.17	lsredis_getd	214
7.7.2.18	lsredis_getl	214
7.7.2.19	lsredis_getstr	214
7.7.2.20	lsredis_hgetCB	215
7.7.2.21	lsredis_init	215
7.7.2.22	lsredis_keysCB	216
7.7.2.23	lsredis_maybe_add_key	217
7.7.2.24	lsredis_regexec	217
7.7.2.25	lsredis_run	217
7.7.2.26	lsredis_set_value	217
7.7.2.27	lsredis_setstr	218
7.7.2.28	lsredis_sig_service	218
7.7.2.29	lsredis_subCB	219
7.7.2.30	lsredis_worker	220
7.7.2.31	redisDisconnectCB	221
7.7.3	Variable Documentation	221
7.7.3.1	lsredis_cond	221
7.7.3.2	lsredis_head	221
7.7.3.3	lsredis_htab	222
7.7.3.4	lsredis_key_select_regex	222
7.7.3.5	lsredis_mutex	222
7.7.3.6	lsredis_objs	222

7.7.3.7	lsredis_publisher	222
7.7.3.8	lsredis_running	222
7.7.3.9	lsredis_thread	222
7.7.3.10	roac	222
7.7.3.11	rofd	222
7.7.3.12	subac	222
7.7.3.13	subfd	222
7.7.3.14	wrac	222
7.7.3.15	wrfd	223
7.8	lstimer.c File Reference	223
7.8.1	Detailed Description	224
7.8.2	Macro Definition Documentation	224
7.8.2.1	LSTIMER_LIST_LENGTH	224
7.8.2.2	LSTIMER_RESOLUTION_NSECS	224
7.8.3	Typedef Documentation	224
7.8.3.1	lstimer_list_t	224
7.8.4	Function Documentation	224
7.8.4.1	handler	224
7.8.4.2	lstimer_add_timer	225
7.8.4.3	lstimer_init	226
7.8.4.4	lstimer_run	226
7.8.4.5	lstimer_worker	226
7.8.4.6	service_timers	227
7.8.5	Variable Documentation	228
7.8.5.1	lstimer_active_timers	228
7.8.5.2	lstimer_cond	228
7.8.5.3	lstimer_list	228
7.8.5.4	lstimer_mutex	228
7.8.5.5	lstimer_thread	229
7.8.5.6	lstimer_timerid	229
7.8.5.7	new_timer	229
7.9	md2cmds.c File Reference	229
7.9.1	Detailed Description	230
7.9.2	Function Documentation	231
7.9.2.1	md2cmds_action_queue	231
7.9.2.2	md2cmds_action_wait	231
7.9.2.3	md2cmds_center	231
7.9.2.4	md2cmds_collect	232
7.9.2.5	md2cmds_init	234
7.9.2.6	md2cmds_kappaphi_move	235

7.9.2.7	<a href="#">md2cmds_maybe_done_moving_cb</a>	235
7.9.2.8	<a href="#">md2cmds_maybe_rotate_done_cb</a>	235
7.9.2.9	<a href="#">md2cmds_move_prep</a>	236
7.9.2.10	<a href="#">md2cmds_move_prep_done_cb</a>	237
7.9.2.11	<a href="#">md2cmds_move_wait</a>	237
7.9.2.12	<a href="#">md2cmds_moveAbs</a>	238
7.9.2.13	<a href="#">md2cmds_mvcenter_move</a>	239
7.9.2.14	<a href="#">md2cmds_organs_move_presets</a>	239
7.9.2.15	<a href="#">md2cmds_organs_prep</a>	240
7.9.2.16	<a href="#">md2cmds_organs_wait</a>	240
7.9.2.17	<a href="#">md2cmds_phase_change</a>	241
7.9.2.18	<a href="#">md2cmds_prep_axis</a>	242
7.9.2.19	<a href="#">md2cmds_rotate</a>	243
7.9.2.20	<a href="#">md2cmds_rotate_cb</a>	244
7.9.2.21	<a href="#">md2cmds_run</a>	244
7.9.2.22	<a href="#">md2cmds_set_scale_cb</a>	245
7.9.2.23	<a href="#">md2cmds_time_capz_cb</a>	245
7.9.2.24	<a href="#">md2cmds_transfer</a>	245
7.9.2.25	<a href="#">md2cmds_worker</a>	248
7.9.3	<a href="#">Variable Documentation</a>	248
7.9.3.1	<a href="#">md2cmds_capz_moving_time</a>	248
7.9.3.2	<a href="#">md2cmds_cmd</a>	249
7.9.3.3	<a href="#">md2cmds_cond</a>	249
7.9.3.4	<a href="#">md2cmds_md_status_code</a>	249
7.9.3.5	<a href="#">md2cmds_moving_cond</a>	249
7.9.3.6	<a href="#">md2cmds_moving_count</a>	249
7.9.3.7	<a href="#">md2cmds_moving_mutex</a>	249
7.9.3.8	<a href="#">md2cmds_moving_pq</a>	249
7.9.3.9	<a href="#">md2cmds_moving_queue_wait</a>	249
7.9.3.10	<a href="#">md2cmds_mutex</a>	249
7.9.3.11	<a href="#">md2cmds_thread</a>	249
7.9.3.12	<a href="#">rotating</a>	250
7.10	<a href="#">mk_pgpmac_redis.py File Reference</a>	250
7.11	<a href="#">pgpmac.c File Reference</a>	250
7.11.1	<a href="#">Detailed Description</a>	251
7.11.2	<a href="#">Function Documentation</a>	251
7.11.2.1	<a href="#">main</a>	251
7.11.2.2	<a href="#">pgpmac_printf</a>	253
7.11.2.3	<a href="#">stdinService</a>	253
7.11.3	<a href="#">Variable Documentation</a>	254

7.11.3.1	<a href="#">ncurses_mutex</a>	254
7.11.3.2	<a href="#">running</a>	255
7.11.3.3	<a href="#">stdinfda</a>	255
7.11.3.4	<a href="#">term_input</a>	255
7.11.3.5	<a href="#">term_output</a>	255
7.11.3.6	<a href="#">term_status</a>	255
7.11.3.7	<a href="#">term_status2</a>	255
7.12	<a href="#">pgpmac.h File Reference</a>	255
7.12.1	<a href="#">Detailed Description</a>	263
7.12.2	<a href="#">Macro Definition Documentation</a>	263
7.12.2.1	<a href="#">_GNU_SOURCE</a>	263
7.12.2.2	<a href="#">LS_DISPLAY_WINDOW_HEIGHT</a>	263
7.12.2.3	<a href="#">LS_DISPLAY_WINDOW_WIDTH</a>	263
7.12.2.4	<a href="#">LS_PG_QUERY_STRING_LENGTH</a>	263
7.12.2.5	<a href="#">LSEVENTS_EVENT_LENGTH</a>	263
7.12.2.6	<a href="#">MD2CMDS_CMD_LENGTH</a>	263
7.12.3	<a href="#">Typedef Documentation</a>	264
7.12.3.1	<a href="#">lspg_demandairrights_t</a>	264
7.12.3.2	<a href="#">lspg_getcenter_t</a>	264
7.12.3.3	<a href="#">lspg_getcurrentsampleid_t</a>	264
7.12.3.4	<a href="#">lspg_nextsample_t</a>	264
7.12.3.5	<a href="#">lspg_nextshot_t</a>	264
7.12.3.6	<a href="#">lspg_query_queue_t</a>	264
7.12.3.7	<a href="#">lspg_starttransfer_t</a>	264
7.12.3.8	<a href="#">lspg_waitcryo_t</a>	264
7.12.3.9	<a href="#">lspmac_bi_t</a>	264
7.12.3.10	<a href="#">lspmac_motor_t</a>	264
7.12.3.11	<a href="#">lsredis_obj_t</a>	264
7.12.3.12	<a href="#">pmac_cmd_queue_t</a>	264
7.12.3.13	<a href="#">pmac_cmd_t</a>	265
7.12.4	<a href="#">Function Documentation</a>	265
7.12.4.1	<a href="#">lsevents_add_listener</a>	265
7.12.4.2	<a href="#">lsevents_init</a>	265
7.12.4.3	<a href="#">lsevents_remove_listener</a>	266
7.12.4.4	<a href="#">lsevents_run</a>	266
7.12.4.5	<a href="#">lsevents_send_event</a>	266
7.12.4.6	<a href="#">lslogging_init</a>	267
7.12.4.7	<a href="#">lslogging_log_message</a>	267
7.12.4.8	<a href="#">lslogging_run</a>	268
7.12.4.9	<a href="#">lspg_array2ptrs</a>	268

7.12.4.10 lspg_demandairrights_all . . . . .	269
7.12.4.11 lspg_getcenter_call . . . . .	270
7.12.4.12 lspg_getcenter_done . . . . .	270
7.12.4.13 lspg_getcenter_wait . . . . .	270
7.12.4.14 lspg_getcurrentsampleid_wait_for_id . . . . .	270
7.12.4.15 lspg_init . . . . .	271
7.12.4.16 lspg_nextsample_all . . . . .	271
7.12.4.17 lspg_nextshot_call . . . . .	271
7.12.4.18 lspg_nextshot_done . . . . .	271
7.12.4.19 lspg_nextshot_wait . . . . .	272
7.12.4.20 lspg_query_push . . . . .	272
7.12.4.21 lspg_run . . . . .	272
7.12.4.22 lspg_seq_run_prep_all . . . . .	273
7.12.4.23 lspg_starttransfer_call . . . . .	273
7.12.4.24 lspg_starttransfer_done . . . . .	273
7.12.4.25 lspg_starttransfer_wait . . . . .	273
7.12.4.26 lspg_waitcryo_all . . . . .	274
7.12.4.27 lspg_waitcryo_cb . . . . .	274
7.12.4.28 lspg_zoom_lut_call . . . . .	274
7.12.4.29 lspmac_getBIPosition . . . . .	274
7.12.4.30 lspmac_getPosition . . . . .	274
7.12.4.31 lspmac_home1_queue . . . . .	275
7.12.4.32 lspmac_init . . . . .	276
7.12.4.33 lspmac_jogabs_queue . . . . .	279
7.12.4.34 lspmac_move_or_jog_abs_queue . . . . .	279
7.12.4.35 lspmac_move_or_jog_preset_queue . . . . .	281
7.12.4.36 lspmac_move_or_jog_queue . . . . .	281
7.12.4.37 lspmac_move_preset_queue . . . . .	281
7.12.4.38 lspmac_moveabs_queue . . . . .	282
7.12.4.39 lspmac_moveabs_wait . . . . .	282
7.12.4.40 lspmac_run . . . . .	283
7.12.4.41 lspmac_SockSendDPLine . . . . .	284
7.12.4.42 lspmac_SockSendline . . . . .	284
7.12.4.43 lspmac_video_rotate . . . . .	285
7.12.4.44 lsredis_cmpnstr . . . . .	285
7.12.4.45 lsredis_cmpstr . . . . .	285
7.12.4.46 lsredis_find_preset . . . . .	286
7.12.4.47 lsredis_get_obj . . . . .	286
7.12.4.48 lsredis_get_string_array . . . . .	287
7.12.4.49 lsredis_getb . . . . .	287

7.12.4.50	lsredis_getd	287
7.12.4.51	lsredis_getl	288
7.12.4.52	lsredis_getstr	288
7.12.4.53	lsredis_init	288
7.12.4.54	lsredis_regexec	289
7.12.4.55	lsredis_run	290
7.12.4.56	lsredis_setstr	290
7.12.4.57	lstimer_add_timer	291
7.12.4.58	lstimer_init	291
7.12.4.59	lstimer_run	292
7.12.4.60	lsupdate_init	292
7.12.4.61	lsupdate_run	292
7.12.4.62	md2cmds_init	292
7.12.4.63	md2cmds_run	292
7.12.4.64	pgpmac_printf	293
7.12.4.65	PmacSockSendline	293
7.12.5	Variable Documentation	293
7.12.5.1	alignx	293
7.12.5.2	aligny	293
7.12.5.3	alignz	293
7.12.5.4	anal	293
7.12.5.5	apery	293
7.12.5.6	aperz	294
7.12.5.7	arm_parked	294
7.12.5.8	blight	294
7.12.5.9	blight_down	294
7.12.5.10	blight_f	294
7.12.5.11	blight_ud	294
7.12.5.12	blight_up	294
7.12.5.13	capy	294
7.12.5.14	capz	294
7.12.5.15	cenx	295
7.12.5.16	ceny	295
7.12.5.17	cryo	295
7.12.5.18	cryo_back	295
7.12.5.19	cryo_switch	295
7.12.5.20	dryer	295
7.12.5.21	etel_init_ok	295
7.12.5.22	etel_on	295
7.12.5.23	etel_ready	295

7.12.5.24 flight	296
7.12.5.25 flight_f	296
7.12.5.26 flight_oo	296
7.12.5.27 fluo	296
7.12.5.28 fluor_back	296
7.12.5.29 fscint	296
7.12.5.30 fshut	296
7.12.5.31 hp_air	296
7.12.5.32 kappa	296
7.12.5.33 lp_air	297
7.12.5.34 lspg_demandairrights	297
7.12.5.35 lspg_getcenter	297
7.12.5.36 lspg_getcurrentsampleid	297
7.12.5.37 lspg_nextsample	297
7.12.5.38 lspg_nextshot	297
7.12.5.39 lspg_starttransfer	297
7.12.5.40 lspg_waitcryo	297
7.12.5.41 lspmac_motors	297
7.12.5.42 lspmac_moving_cond	298
7.12.5.43 lspmac_moving_flags	298
7.12.5.44 lspmac_moving_mutex	298
7.12.5.45 lspmac_nmotors	298
7.12.5.46 lspmac_shutter_cond	298
7.12.5.47 lspmac_shutter_has_opened	298
7.12.5.48 lspmac_shutter_mutex	298
7.12.5.49 lspmac_shutter_state	298
7.12.5.50 md2_status_mutex	298
7.12.5.51 md2cmds_cmd	299
7.12.5.52 md2cmds_cond	299
7.12.5.53 md2cmds_md_status_code	299
7.12.5.54 md2cmds_mutex	299
7.12.5.55 md2cmds_pg_cond	299
7.12.5.56 md2cmds_pg_mutex	299
7.12.5.57 minikappa_ok	299
7.12.5.58 ncurses_mutex	299
7.12.5.59 omega	299
7.12.5.60 omega_zero_time	299
7.12.5.61 phi	299
7.12.5.62 pmac_queue_cond	300
7.12.5.63 pmac_queue_mutex	300

7.12.5.64 sample_detected . . . . .	300
7.12.5.65 scint . . . . .	300
7.12.5.66 shutter_open . . . . .	300
7.12.5.67 smart_mag_err . . . . .	300
7.12.5.68 smart_mag_off . . . . .	300
7.12.5.69 smart_mag_on . . . . .	300
7.12.5.70 smart_mag_oo . . . . .	300
7.12.5.71 term_input . . . . .	301
7.12.5.72 term_output . . . . .	301
7.12.5.73 term_status . . . . .	301
7.12.5.74 term_status2 . . . . .	301
7.12.5.75 zoom . . . . .	301

**Index****301**



# 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 effecient 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:

<a href="#">lsevents.c</a>	Simple event queue
<a href="#">lsredis.c</a>	Receive key value pair updates from redis databases
<a href="#">lslogging.c</a>	A logging utility to simplify debugging
<a href="#">lspg.c</a>	Handles communications with the controlling posgresql database
<a href="#">lsupdate.c</a>	Periodically update the px.kvs table with new positions.
<a href="#">md2cmds.c</a>	Provides the equivilant (mostly) of the LS-CAT BLUMax code.
<a href="#">pgpmac.c</a>	Main: parses command line and starts up the various threads
<a href="#">pgpmac.h</a>	All includes and defines. The only file included by the .c files in this
<a href="#">pmac_md2_ls-cat.pmc</a>	Code for the PMAC: compile and install with pmac exectutive program.
<a href="#">pmac_md2.sql</a>	Tables and procedures for the posgresql 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 pgp-mac 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 . . . . .	15
<a href="#">lsevents_listener_struct</a>	Linked list of event listeners . . . . .	17
<a href="#">lsevents_queue_struct</a>	Storage definition for the events . . . . .	18
<a href="#">lslogging_queue_struct</a>	Our log object: time and message . . . . .	19
<a href="#">lspg_demandairrights_struct</a>	. . . . .	20
<a href="#">lspg_getcenter_struct</a>	Storage for getcenter query Used for the md2 ROTATE command that generates the centering movies . . . . .	20
<a href="#">lspg_getcurrentsampleid_struct</a>	. . . . .	23
<a href="#">lspg_lock_detector_struct</a>	Lock detector object Implements detector lock for exposure control . . . . .	24
<a href="#">lspg_lock_diffractionmeter_struct</a>	Object used to impliment locking the diffractometer Critical to exposure timing . . . . .	25
<a href="#">lspg_nextsample_struct</a>	Returns the next sample number Just a 32 bit int (Ha!, take that, nextshot!) . . . . .	25
<a href="#">lspg_nextshot_struct</a>	Storage definition for nextshot query . . . . .	27
<a href="#">lspg_seq_run_prep_struct</a>	Data collection running object . . . . .	37
<a href="#">lspg_starttransfer_struct</a>	Returns 1 if transfer can continue 0 to abort . . . . .	38
<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 . . . . .	39
<a href="#">lspg_waitcryo_struct</a>	. . . . .	40
<a href="#">lspgQueryQueueStruct</a>	Store each query along with it's callback function . . . . .	41
<a href="#">lspmac_asci_buffers_struct</a>	. . . . .	42
<a href="#">lspmac_bi_struct</a>	Storage for binary inputs . . . . .	42
<a href="#">lspmac_cmd_queue_struct</a>	PMAC command queue item . . . . .	44

<a href="#">lspmac_dpascii_queue_struct</a>	45
<a href="#">lspmac_motor_struct</a>	
Motor information	46
<a href="#">lsredis_obj_struct</a>	
Redis Object Basic object whose value is synchronized with our redis db	52
<a href="#">lstimer_list_struct</a>	
Everything we need to know about a timer	55
<a href="#">md2StatusStruct</a>	
The block of memory retrieved in a status request	57
<a href="#">tagEthernetCmd</a>	
PMAC ethernet packet definition	64



## Chapter 4

# File Index

### 4.1 File List

Here is a list of all files with brief descriptions:

<a href="#">iniParser.py</a>	67
<a href="#">kvredis.c</a>	67
<a href="#">lsevents.c</a>	
Event subsystem for inter-pgpmac communication	83
<a href="#">lslogging.c</a>	
Logs messages to a file	89
<a href="#">lspg.c</a>	
Postgresql support for the LS-CAT pgpmac project	93
<a href="#">lspmac.c</a>	
Routines concerned with communication with PMAC	133
<a href="#">lsredis.c</a>	
Support redis hash synchronization	204
<a href="#">lstimer.c</a>	
Support for delayed and periodic events	223
<a href="#">md2cmds.c</a>	
Implements commands to run the md2 diffractometer attached to a PMAC controled by post- gresql	229
<a href="#">mk_pgpmac_redis.py</a>	250
<a href="#">pgpmac.c</a>	
Main for the pgpmac project	250
<a href="#">pgpmac.h</a>	
Headers for the entire pgpmac project	255



## 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](#)( "microdiff\_hard.ini")

#### 5.1.1 Variable Documentation

##### 5.1.1.1 tuple [iniParser.ip](#) [iniParser](#)( "microdiff\_hard.ini")

Definition at line 104 of file [iniParser.py](#).

### 5.2 mk\_pgpmac\_redis Namespace Reference

#### Variables

- list [head](#) sys.argv[1]
- list [pref\\_ini](#) sys.argv[2]
- list [hard\\_ini](#) sys.argv[3]
- 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"
- 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")`

## 5.2.1 Variable Documentation

### 5.2.1.1 tuple `mk_pgpmac_redis.b` `pi.get( section, "LightIntensity")`

Definition at line 376 of file `mk_pgpmac_redis.py`.

### 5.2.1.2 list `mk_pgpmac_redis.bi_list` `["CryoSwitch"]`

Definition at line 226 of file `mk_pgpmac_redis.py`.

### 5.2.1.3 tuple `mk_pgpmac_redis.f` `"HSETNX"`

Definition at line 326 of file `mk_pgpmac_redis.py`.

### 5.2.1.4 string `mk_pgpmac_redis.fnc` `"HSETNX"`

Definition at line 368 of file `mk_pgpmac_redis.py`.

### 5.2.1.5 `mk_pgpmac_redis.hard_ini` `sys.argv[3]`

Definition at line 21 of file `mk_pgpmac_redis.py`.

### 5.2.1.6 dictionary `mk_pgpmac_redis.hard_ini_fields`

**Initial value:**

```

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

Definition at line 169 of file `mk_pgpmac_redis.py`.

### 5.2.1.7 list `mk_pgpmac_redis.head` `sys.argv[1]`

Definition at line 13 of file `mk_pgpmac_redis.py`.

### 5.2.1.8 tuple `mk_pgpmac_redis.hi` `iniParser.iniParser( hard_ini)`

Definition at line 301 of file `mk_pgpmac_redis.py`.

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

Definition at line 347 of file mk\_pgpmac\_redis.py.

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

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

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

Definition at line 190 of file mk\_pgpmac\_redis.py.

#### 5.2.1.12 dictionary mk\_pgpmac\_redis.motor\_presets

Definition at line 228 of file mk\_pgpmac\_redis.py.

#### 5.2.1.13 tuple mk\_pgpmac\_redis.p pi.get( section, "MotorPosition")

Definition at line 382 of file mk\_pgpmac\_redis.py.

#### 5.2.1.14 tuple mk\_pgpmac\_redis.pi iniParser.iniParser( pref\_ini)

Definition at line 344 of file mk\_pgpmac\_redis.py.

#### 5.2.1.15 tuple mk\_pgpmac\_redis.ppos pi.get( section, option)

Definition at line 354 of file mk\_pgpmac\_redis.py.

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

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

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

Definition at line 325 of file mk\_pgpmac\_redis.py.

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

Definition at line 388 of file mk\_pgpmac\_redis.py.

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

Definition at line 394 of file mk\_pgpmac\_redis.py.

#### 5.2.1.20 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 277 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
81         self.f.close()
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 Isevents\_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*

### 6.2.1 Detailed Description

Linked list of event listeners.

Definition at line 27 of file lsevents.c.

### 6.2.2 Field Documentation

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

call back function

Definition at line 31 of file lsevents.c.

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

Next listener.

Definition at line 28 of file lsevents.c.

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

the original string sent to us

Definition at line 29 of file lsevents.c.

#### 6.2.2.4 regex\_t lsevents\_listener\_struct::re

regular expression representing listened for events

Definition at line 30 of file lsevents.c.

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

- [lsevents.c](#)

## 6.3 lsevents\_queue\_struct Struct Reference

Storage definition for the events.

## Data Fields

- char \* [evp](#)  
*name of the event*

### 6.3.1 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.

### 6.3.2 Field Documentation

#### 6.3.2.1 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](#)

## 6.4 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*

### 6.4.1 Detailed Description

Our log object: time and message.

Definition at line 24 of file lslogging.c.

### 6.4.2 Field Documentation

#### 6.4.2.1 char lslogging\_queue\_struct::lmsg[LSLOGGING\_MSG\_LENGTH]

our message, truncated if too long

Definition at line 26 of file lslogging.c.

#### 6.4.2.2 struct timespec lslogging\_queue\_struct::itime

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](#)

## 6.5 lspg\_demandairrights\_struct Struct Reference

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

### Data Fields

- pthread\_mutex\_t [mutex](#)
- pthread\_cond\_t [cond](#)
- int [new\\_value\\_ready](#)

#### 6.5.1 Detailed Description

Definition at line 190 of file pgpmac.h.

#### 6.5.2 Field Documentation

##### 6.5.2.1 pthread\_cond\_t lspg\_demandairrights\_struct::cond

Definition at line 192 of file pgpmac.h.

##### 6.5.2.2 pthread\_mutex\_t lspg\_demandairrights\_struct::mutex

Definition at line 191 of file pgpmac.h.

##### 6.5.2.3 int lspg\_demandairrights\_struct::new\_value\_ready

Definition at line 193 of file pgpmac.h.

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

- [pgpmac.h](#)

## 6.6 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](#)

### 6.6.1 Detailed Description

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

Definition at line 204 of file pgpmac.h.

### 6.6.2 Field Documentation

#### 6.6.2.1 pthread\_cond\_t lspg\_getcenter\_struct::cond

provides signaling for when the query is done

Definition at line 206 of file pgpmac.h.

#### 6.6.2.2 double lspg\_getcenter\_struct::dax

alignment x change

Definition at line 219 of file pgpmac.h.

#### 6.6.2.3 int lspg\_getcenter\_struct::dax\_isnull

Definition at line 220 of file pgpmac.h.

#### 6.6.2.4 `double lspg_getcenter_struct::day`

alignment y change

Definition at line 222 of file pgpmac.h.

#### 6.6.2.5 `int lspg_getcenter_struct::day_isnull`

Definition at line 223 of file pgpmac.h.

#### 6.6.2.6 `double lspg_getcenter_struct::daz`

alignment z change

Definition at line 225 of file pgpmac.h.

#### 6.6.2.7 `int lspg_getcenter_struct::daz_isnull`

Definition at line 226 of file pgpmac.h.

#### 6.6.2.8 `double lspg_getcenter_struct::dcx`

center x change

Definition at line 213 of file pgpmac.h.

#### 6.6.2.9 `int lspg_getcenter_struct::dcx_isnull`

Definition at line 214 of file pgpmac.h.

#### 6.6.2.10 `double lspg_getcenter_struct::dcy`

center y change

Definition at line 216 of file pgpmac.h.

#### 6.6.2.11 `int lspg_getcenter_struct::dcy_isnull`

Definition at line 217 of file pgpmac.h.

#### 6.6.2.12 `pthread_mutex_t lspg_getcenter_struct::mutex`

don't let the threads collide!

Definition at line 205 of file pgpmac.h.

#### 6.6.2.13 `int lspg_getcenter_struct::new_value_ready`

used with condition

Definition at line 207 of file pgpmac.h.

#### 6.6.2.14 int lspg\_getcenter\_struct::no\_rows\_returned

flag in case no centering information was forthcoming

Definition at line 208 of file pgpmac.h.

#### 6.6.2.15 int lspg\_getcenter\_struct::zoom

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

Definition at line 210 of file pgpmac.h.

#### 6.6.2.16 int lspg\_getcenter\_struct::zoom\_isnull

Definition at line 211 of file pgpmac.h.

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

- [pgpmac.h](#)

## 6.7 lspg\_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*

### 6.7.1 Detailed Description

Definition at line 178 of file pgpmac.h.

### 6.7.2 Field Documentation

#### 6.7.2.1 pthread\_cond\_t lspg\_getcurrentsampleid\_struct::cond

for signaling

Definition at line 180 of file pgpmac.h.

#### 6.7.2.2 unsigned int lspg\_getcurrentsampleid\_struct::getcurrentsampleid

the sample we think is mounted on the diffractometer

Definition at line 183 of file pgpmac.h.

#### 6.7.2.3 int lspg\_getcurrentsampleid\_struct::getcurrentsampleid\_isnull

the sample we think is mounted on the diffractometer

Definition at line 184 of file pgpmac.h.

#### 6.7.2.4 pthread\_mutex\_t lspg\_getcurrentsampleid\_struct::mutex

practice safe threading

Definition at line 179 of file pgpmac.h.

#### 6.7.2.5 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 182 of file pgpmac.h.

#### 6.7.2.6 int lspg\_getcurrentsampleid\_struct::no\_rows\_returned

flag for an empty return

Definition at line 181 of file pgpmac.h.

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

- [pgpmac.h](#)

## 6.8 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](#)

### 6.8.1 Detailed Description

lock detector object Implements detector lock for exposure control

Definition at line 974 of file lspg.c.

### 6.8.2 Field Documentation

#### 6.8.2.1 pthread\_cond\_t lspg\_lock\_detector\_struct::cond

Definition at line 976 of file lspg.c.



#### 6.8.2.2 pthread\_mutex\_t lspg\_lock\_detector\_struct::mutex

Definition at line 975 of file lspg.c.

#### 6.8.2.3 int lspg\_lock\_detector\_struct::new\_value\_ready

Definition at line 977 of file lspg.c.

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

- [lspg.c](#)

## 6.9 lspg\_lock\_diffractionmeter\_struct Struct Reference

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

### Data Fields

- pthread\_mutex\_t [mutex](#)
- pthread\_cond\_t [cond](#)
- int [new\\_value\\_ready](#)

### 6.9.1 Detailed Description

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

Definition at line 915 of file lspg.c.

### 6.9.2 Field Documentation

#### 6.9.2.1 pthread\_cond\_t lspg\_lock\_diffractionmeter\_struct::cond

Definition at line 917 of file lspg.c.

#### 6.9.2.2 pthread\_mutex\_t lspg\_lock\_diffractionmeter\_struct::mutex

Definition at line 916 of file lspg.c.

#### 6.9.2.3 int lspg\_lock\_diffractionmeter\_struct::new\_value\_ready

Definition at line 918 of file lspg.c.

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

- [lspg.c](#)

## 6.10 lspg\_nextsample\_struct Struct Reference

Returns the next sample number Just a 32 bit int (Ha!, 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.*

### 6.10.1 Detailed Description

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

Definition at line 251 of file pgpmac.h.

### 6.10.2 Field Documentation

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

Our condition.

Definition at line 253 of file pgpmac.h.

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

Our mutex.

Definition at line 252 of file pgpmac.h.

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

flag for our condition

Definition at line 254 of file pgpmac.h.

#### 6.10.2.4 unsigned int lspg\_nextsample\_struct::nextsample

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

Definition at line 257 of file pgpmac.h.

#### 6.10.2.5 int lspg\_nextsample\_struct::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.

Definition at line 258 of file pgpmac.h.

## 6.10.2.6 int lspg\_nextsample\_struct::no\_rows\_returned

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

Definition at line 255 of file pgpmac.h.

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

- [pgpmac.h](#)

## 6.11 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 ascillation 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](#)

### 6.11.1 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 271 of file pgpmac.h.

### 6.11.2 Field Documentation

#### 6.11.2.1 int lspg\_nextshot\_struct::active

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

Definition at line 334 of file pgpmac.h.

#### 6.11.2.2 `int lspg_nextshot_struct::active2`

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

Definition at line 385 of file pgpmac.h.

#### 6.11.2.3 `int lspg_nextshot_struct::active2_isnull`

Definition at line 386 of file pgpmac.h.

#### 6.11.2.4 `int lspg_nextshot_struct::active_isnull`

Definition at line 335 of file pgpmac.h.

#### 6.11.2.5 `double lspg_nextshot_struct::ax`

alignment table x position

Definition at line 325 of file pgpmac.h.

#### 6.11.2.6 `double lspg_nextshot_struct::ax2`

next image alignment x position

Definition at line 376 of file pgpmac.h.

#### 6.11.2.7 `int lspg_nextshot_struct::ax2_isnull`

Definition at line 377 of file pgpmac.h.

#### 6.11.2.8 `int lspg_nextshot_struct::ax_isnull`

Definition at line 326 of file pgpmac.h.

#### 6.11.2.9 `double lspg_nextshot_struct::ay`

alignment table y position

Definition at line 328 of file pgpmac.h.

#### 6.11.2.10 `double lspg_nextshot_struct::ay2`

next image alignment y position

Definition at line 379 of file pgpmac.h.

#### 6.11.2.11 `int lspg_nextshot_struct::ay2_isnull`

Definition at line 380 of file pgpmac.h.

#### 6.11.2.12 `int lspg_nextshot_struct::ay_isnull`

Definition at line 329 of file pgpmac.h.

**6.11.2.13 double lspg\_nextshot\_struct::az**

alignment table z position

Definition at line 331 of file pgpmac.h.

**6.11.2.14 double lspg\_nextshot\_struct::az2**

next image alignment z position

Definition at line 382 of file pgpmac.h.

**6.11.2.15 int lspg\_nextshot\_struct::az2\_isnull**

Definition at line 383 of file pgpmac.h.

**6.11.2.16 int lspg\_nextshot\_struct::az\_isnull**

Definition at line 332 of file pgpmac.h.

**6.11.2.17 pthread\_cond\_t lspg\_nextshot\_struct::cond**

Condition to wait for a response from our postgresql server.

Definition at line 273 of file pgpmac.h.

**6.11.2.18 double lspg\_nextshot\_struct::cx**

centering table x position

Definition at line 319 of file pgpmac.h.

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

next image centering table x position

Definition at line 370 of file pgpmac.h.

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

Definition at line 371 of file pgpmac.h.

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

Definition at line 320 of file pgpmac.h.

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

centering table y position

Definition at line 322 of file pgpmac.h.

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

next image centering table y position

Definition at line 373 of file pgpmac.h.

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

Definition at line 374 of file pgpmac.h.

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

Definition at line 323 of file pgpmac.h.

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

Directory for data relative to the ESAF home directory.

Definition at line 277 of file pgpmac.h.

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

Definition at line 278 of file pgpmac.h.

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

dataset defined detector distance

Definition at line 310 of file pgpmac.h.

**6.11.2.29 double lspg\_nextshot\_struct::dsdist2**

next image distance

Definition at line 364 of file pgpmac.h.

**6.11.2.30 int lspg\_nextshot\_struct::dsdist2\_isnull**

Definition at line 365 of file pgpmac.h.

**6.11.2.31 int lspg\_nextshot\_struct::dsdist\_isnull**

Definition at line 311 of file pgpmac.h.

**6.11.2.32 double lspg\_nextshot\_struct::dsexp**

dataset defined exposure time

Definition at line 289 of file pgpmac.h.



**6.11.2.33** `double lspg_nextshot_struct::dsexp2`

next image exposure time

Definition at line 349 of file `pgpmac.h`.

**6.11.2.34** `int lspg_nextshot_struct::dsexp2_isnull`

Definition at line 350 of file `pgpmac.h`.

**6.11.2.35** `int lspg_nextshot_struct::dsexp_isnull`

Definition at line 290 of file `pgpmac.h`.

**6.11.2.36** `unsigned int lspg_nextshot_struct::dshpid`

sample holder ID

Definition at line 316 of file `pgpmac.h`.

**6.11.2.37** `int lspg_nextshot_struct::dshpid_isnull`

Definition at line 317 of file `pgpmac.h`.

**6.11.2.38** `double lspg_nextshot_struct::dskappa`

dataset defined starting kappa angle

Definition at line 307 of file `pgpmac.h`.

**6.11.2.39** `double lspg_nextshot_struct::dskappa2`

next image kappa position

Definition at line 361 of file `pgpmac.h`.

**6.11.2.40** `int lspg_nextshot_struct::dskappa2_isnull`

Definition at line 362 of file `pgpmac.h`.

**6.11.2.41** `int lspg_nextshot_struct::dskappa_isnull`

Definition at line 308 of file `pgpmac.h`.

**6.11.2.42** `double lspg_nextshot_struct::dsnr`

dataset defined energy

Definition at line 313 of file `pgpmac.h`.

#### 6.11.2.43 `double lspg_nextshot_struct::dsnrg2`

next image energy

Definition at line 367 of file pgpmac.h.

#### 6.11.2.44 `int lspg_nextshot_struct::dsnrg2_isnull`

Definition at line 368 of file pgpmac.h.

#### 6.11.2.45 `int lspg_nextshot_struct::dsnrg_isnull`

Definition at line 314 of file pgpmac.h.

#### 6.11.2.46 `double lspg_nextshot_struct::dsomega`

dataset defined starting omega angle

Definition at line 304 of file pgpmac.h.

#### 6.11.2.47 `double lspg_nextshot_struct::dsomega2`

next image omega position

Definition at line 358 of file pgpmac.h.

#### 6.11.2.48 `int lspg_nextshot_struct::dsomega2_isnull`

Definition at line 359 of file pgpmac.h.

#### 6.11.2.49 `int lspg_nextshot_struct::dsomega_isnull`

Definition at line 305 of file pgpmac.h.

#### 6.11.2.50 `char* lspg_nextshot_struct::dsoscaxis`

dataset defined oscillation axis (always omega)

Definition at line 286 of file pgpmac.h.

#### 6.11.2.51 `char* lspg_nextshot_struct::dsoscaxis2`

next image ascillation axis (always "omega")

Definition at line 346 of file pgpmac.h.

#### 6.11.2.52 `int lspg_nextshot_struct::dsoscaxis2_isnull`

Definition at line 347 of file pgpmac.h.

#### 6.11.2.53 `int lspg_nextshot_struct::dsoscaxis_isnull`

Definition at line 287 of file pgpmac.h.

**6.11.2.54 double lspg\_nextshot\_struct::dsowidth**

dataset defined oscillation width

Definition at line 283 of file pgpmac.h.

**6.11.2.55 double lspg\_nextshot\_struct::dsowidth2**

next image oscillation width

Definition at line 343 of file pgpmac.h.

**6.11.2.56 int lspg\_nextshot\_struct::dsowidth2\_isnull**

Definition at line 344 of file pgpmac.h.

**6.11.2.57 int lspg\_nextshot\_struct::dsowidth\_isnull**

Definition at line 284 of file pgpmac.h.

**6.11.2.58 double lspg\_nextshot\_struct::dsphi**

dataset defined starting phi angle

Definition at line 301 of file pgpmac.h.

**6.11.2.59 double lspg\_nextshot\_struct::dsphi2**

next image phi position

Definition at line 355 of file pgpmac.h.

**6.11.2.60 int lspg\_nextshot\_struct::dsphi2\_isnull**

Definition at line 356 of file pgpmac.h.

**6.11.2.61 int lspg\_nextshot\_struct::dsphi\_isnull**

Definition at line 302 of file pgpmac.h.

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

ID string identifying this dataset.

Definition at line 280 of file pgpmac.h.

**6.11.2.63 int lspg\_nextshot\_struct::dspid\_isnull**

Definition at line 281 of file pgpmac.h.

**6.11.2.64 pthread\_mutex\_t lspg\_nextshot\_struct::mutex**

Our mutex for sanity in the multi-threaded program.

Definition at line 272 of file pgpmac.h.

**6.11.2.65 int lspg\_nextshot\_struct::new\_value\_ready**

Our flag for the condition to wait for.

Definition at line 274 of file pgpmac.h.

**6.11.2.66 int lspg\_nextshot\_struct::no\_rows\_returned**

flag indicating that no rows were returned.

Definition at line 275 of file pgpmac.h.

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

file name

Definition at line 298 of file pgpmac.h.

**6.11.2.68 int lspg\_nextshot\_struct::sfn\_isnull**

Definition at line 299 of file pgpmac.h.

**6.11.2.69 int lspg\_nextshot\_struct::sindex**

index of frame (used to generate the file extension)

Definition at line 337 of file pgpmac.h.

**6.11.2.70 int lspg\_nextshot\_struct::sindex2**

next image index number

Definition at line 388 of file pgpmac.h.

**6.11.2.71 int lspg\_nextshot\_struct::sindex2\_isnull**

Definition at line 389 of file pgpmac.h.

**6.11.2.72 int lspg\_nextshot\_struct::sindex\_isnull**

Definition at line 338 of file pgpmac.h.

**6.11.2.73 long long lspg\_nextshot\_struct::skey**

key identifying a particular image

Definition at line 292 of file pgpmac.h.

6.11.2.74 `int lspg_nextshot_struct::skey_isnull`

Definition at line 293 of file pgpmac.h.

6.11.2.75 `double lspg_nextshot_struct::sstart`

starting angle

Definition at line 295 of file pgpmac.h.

6.11.2.76 `double lspg_nextshot_struct::sstart2`

next image start angle

Definition at line 352 of file pgpmac.h.

6.11.2.77 `int lspg_nextshot_struct::sstart2_isnull`

Definition at line 353 of file pgpmac.h.

6.11.2.78 `int lspg_nextshot_struct::sstart_isnull`

Definition at line 296 of file pgpmac.h.

6.11.2.79 `char* lspg_nextshot_struct::stype`

"Normal" or "Gridsearch"

Definition at line 340 of file pgpmac.h.

6.11.2.80 `char* lspg_nextshot_struct::stype2`

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

Definition at line 391 of file pgpmac.h.

6.11.2.81 `int lspg_nextshot_struct::stype2_isnull`

Definition at line 392 of file pgpmac.h.

6.11.2.82 `int lspg_nextshot_struct::stype_isnull`

Definition at line 341 of file pgpmac.h.

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

- [pgpmac.h](#)

## 6.12 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](#)

### 6.12.1 Detailed Description

Data collection running object.

Definition at line 1032 of file lspg.c.

### 6.12.2 Field Documentation

#### 6.12.2.1 pthread\_cond\_t lspg\_seq\_run\_prep\_struct::cond

Definition at line 1034 of file lspg.c.

#### 6.12.2.2 pthread\_mutex\_t lspg\_seq\_run\_prep\_struct::mutex

Definition at line 1033 of file lspg.c.

#### 6.12.2.3 int lspg\_seq\_run\_prep\_struct::new\_value\_ready

Definition at line 1035 of file lspg.c.

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

- [lspg.c](#)

## 6.13 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)*

### 6.13.1 Detailed Description

returns 1 if transfer can continue 0 to abort

Definition at line 237 of file `pgpmac.h`.

### 6.13.2 Field Documentation

#### 6.13.2.1 `pthread_cond_t lspg_starttransfer_struct::cond`

Our condition.

Definition at line 239 of file `pgpmac.h`.

#### 6.13.2.2 `pthread_mutex_t lspg_starttransfer_struct::mutex`

Our mutex.

Definition at line 238 of file `pgpmac.h`.

#### 6.13.2.3 `int lspg_starttransfer_struct::new_value_ready`

flag for our condition

Definition at line 240 of file `pgpmac.h`.

#### 6.13.2.4 `int lspg_starttransfer_struct::no_rows_returned`

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

Definition at line 241 of file `pgpmac.h`.

#### 6.13.2.5 `unsigned int lspg_starttransfer_struct::starttransfer`

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

Definition at line 243 of file `pgpmac.h`.

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

- [pgpmac.h](#)

## 6.14 `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](#)

### 6.14.1 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 850 of file lspg.c.

### 6.14.2 Field Documentation

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

Definition at line 852 of file lspg.c.

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

Definition at line 851 of file lspg.c.

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

Definition at line 853 of file lspg.c.

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

- [lspg.c](#)

## 6.15 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.*

### 6.15.1 Detailed Description

Definition at line 170 of file pgpmac.h.

### 6.15.2 Field Documentation

#### 6.15.2.1 pthread\_cond\_t lspg\_waitcryo\_struct::cond

for signaling

Definition at line 172 of file pgpmac.h.



## 6.15.2.2 pthread\_mutex\_t Ispg\_waitcryo\_struct::mutex

practice safe threading

Definition at line 171 of file pgpmac.h.

## 6.15.2.3 int Ispg\_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 173 of file pgpmac.h.

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

- [pgpmac.h](#)

## 6.16 IspgQueryQueueStruct Struct Reference

Store each query along with it's 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 [IspgQueryQueueStruct](#) \*qq, PGresult \*pgr)  
*Callback function for when a query returns a result.*

### 6.16.1 Detailed Description

Store each query along with it's callback function.

All calls are asynchronous

Definition at line 31 of file kvredis.c.

### 6.16.2 Field Documentation

## 6.16.2.1 void(\* IspgQueryQueueStruct::onResponse)(struct IspgQueryQueueStruct \*qq, PGresult \*pgr)

Callback function for when a query returns a result.

Definition at line 33 of file kvredis.c.

## 6.16.2.2 char IspgQueryQueueStruct::qs

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

Definition at line 32 of file kvredis.c.

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

- [kvredis.c](#)
- [pgpmac.h](#)

## 6.17 Ispmac\_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]

### 6.17.1 Detailed Description

Definition at line 342 of file `Ispmac.c`.

### 6.17.2 Field Documentation

#### 6.17.2.1 uint16\_t Ispmac\_ascii\_buffers\_struct::command\_buf

Definition at line 344 of file `Ispmac.c`.

#### 6.17.2.2 uint16\_t Ispmac\_ascii\_buffers\_struct::command\_buf\_cc

Definition at line 345 of file `Ispmac.c`.

#### 6.17.2.3 char Ispmac\_ascii\_buffers\_struct::command\_str[160]

Definition at line 346 of file `Ispmac.c`.

#### 6.17.2.4 uint16\_t Ispmac\_ascii\_buffers\_struct::response\_buf

Definition at line 347 of file `Ispmac.c`.

#### 6.17.2.5 uint16\_t Ispmac\_ascii\_buffers\_struct::response\_n

Definition at line 348 of file `Ispmac.c`.

#### 6.17.2.6 char Ispmac\_ascii\_buffers\_struct::response\_str[256]

Definition at line 349 of file `Ispmac.c`.

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

- [Ispmac.c](#)

## 6.18 Ispmac\_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.*

### 6.18.1 Detailed Description

Storage for binary inputs.

Definition at line 150 of file pgpmac.h.

### 6.18.2 Field Documentation

#### 6.18.2.1 char\* Ispmac\_bi\_struct::changeEventOff

Event to send when the value changes to 0.

Definition at line 158 of file pgpmac.h.

#### 6.18.2.2 char\* Ispmac\_bi\_struct::changeEventOn

Event to send when the value changes to 1.

Definition at line 157 of file pgpmac.h.

#### 6.18.2.3 int Ispmac\_bi\_struct::first\_time

flag indicating we've not read the input even once

Definition at line 156 of file pgpmac.h.

#### 6.18.2.4 int Ispmac\_bi\_struct::mask

mask for the bit in the status register

Definition at line 153 of file pgpmac.h.

#### 6.18.2.5 pthread\_mutex\_t lspmact::mutex

so we don't get confused

Definition at line 152 of file pgpmac.h.

#### 6.18.2.6 int lspmact::position

the current value.

Definition at line 154 of file pgpmac.h.

#### 6.18.2.7 int lspmact::previous

the previous value

Definition at line 155 of file pgpmac.h.

#### 6.18.2.8 int\* lspmact::ptr

points to the location in the status buffer

Definition at line 151 of file pgpmac.h.

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

- [pgpmac.h](#)

## 6.19 lspmact\_cmd\_queue\_struct Struct Reference

PMAC command queue item.

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

### Data Fields

- [pmact\\_cmd\\_t pcmd](#)  
*the pmact 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 pmact*
- char \* [event](#)  
*event name to send*
- void(\* [onResponse](#))(struct [lspmact\\_cmd\\_queue\\_struct](#) \*, int, char \*)  
*function to call when response is received. args are (int fd, nreturned, buffer)*

### 6.19.1 Detailed Description

PMAC command queue item.

Command queue items are fixed length to simplify memory management.

Definition at line 86 of file pgpmac.h.

### 6.19.2 Field Documentation

#### 6.19.2.1 `char* lspmac_cmd_queue_struct::event`

event name to send

Definition at line 90 of file `pgpmac.h`.

#### 6.19.2.2 `int lspmac_cmd_queue_struct::no_reply`

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

Definition at line 88 of file `pgpmac.h`.

#### 6.19.2.3 `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`.

#### 6.19.2.4 `pmac_cmd_t lspmac_cmd_queue_struct::pcmd`

the pmac command to send

Definition at line 87 of file `pgpmac.h`.

#### 6.19.2.5 `struct timespec lspmac_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](#)

## 6.20 `lspmac_dpascii_queue_struct` Struct Reference

### Data Fields

- `char * event`
- `char pl[160]`

### 6.20.1 Detailed Description

Definition at line 356 of file `lspmac.c`.

### 6.20.2 Field Documentation

#### 6.20.2.1 `char* lspmac_dpascii_queue_struct::event`

Definition at line 357 of file `lspmac.c`.

### 6.20.2.2 char lspmac\_dpascii\_queue\_struct::pl[160]

Definition at line 358 of file lspmac.c.

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

- [lspmac.c](#)

## 6.21 lspmac\_motor\_struct Struct Reference

Motor information.

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

### Data Fields

- 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 [homing](#)  
*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](#)  
*scaled position*
- double [reported\\_position](#)  
*previous position reported to the database*
- 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](#)  
*Second 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 referred by ls database kvs table.*
- [lsredis\\_obj\\_t](#) \* [unit](#)  
*string to use as the units*
- [lsredis\\_obj\\_t](#) \* [printf\\_fmt](#)  
*printf format*
- [lsredis\\_obj\\_t](#) \* [redis\\_fmt](#)  
*special format string to create text array for putting the position back into redis*
- [lsredis\\_obj\\_t](#) \* [max\\_speed](#)  
*our maximum speed (cts/msec)*
- [lsredis\\_obj\\_t](#) \* [max\\_accel](#)  
*our maximum acceleration (cts/msec<sup>2</sup>)*
- [lsredis\\_obj\\_t](#) \* [motor\\_num](#)  
*pmac motor number*
- [lsredis\\_obj\\_t](#) \* [coord\\_num](#)  
*coordinate system this motor belongs to (0 if none)*
- [lsredis\\_obj\\_t](#) \* [update\\_resolution](#)  
*Change needs to be at least this big to report as a new position to the database.*
- [lsredis\\_obj\\_t](#) \* [axis](#)  
*the axis (X, Y, Z, etc) or null if not in a coordinate system*
- [lsredis\\_obj\\_t](#) \* [home](#)  
*pmac commands to home motor*
- [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](#) \* [inactive\\_init](#)  
*pmac commands to inactivate the motor*
- [lsredis\\_obj\\_t](#) \* [redis\\_position](#)  
*how we report our position to the world*
- [lsredis\\_obj\\_t](#) \* [status\\_str](#)  
*A talky version of the status.*
- [lsredis\\_obj\\_t](#) \* [u2c](#)  
*conversion from counts to units: 0.0 means not loaded yet*
- 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.*
- void(\* [moveAbs](#))(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*

### 6.21.1 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.

### 6.21.2 Field Documentation

#### 6.21.2.1 `Isredis_obj_t* lspmac_motor_struct::active`

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

Definition at line 132 of file pgpmac.h.

#### 6.21.2.2 `Isredis_obj_t* lspmac_motor_struct::active_init`

pmac commands to make this motor active

Definition at line 133 of file pgpmac.h.

#### 6.21.2.3 `int lspmac_motor_struct::actual_pos_cnts`

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

Definition at line 112 of file pgpmac.h.

#### 6.21.2.4 `int* lspmac_motor_struct::actual_pos_cnts_p`

pointer to the md2\_status structure to the actual position

Definition at line 111 of file pgpmac.h.

#### 6.21.2.5 `Isredis_obj_t* lspmac_motor_struct::axis`

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

Definition at line 130 of file pgpmac.h.

#### 6.21.2.6 `int lspmac_motor_struct::command_sent`

Motion command verified sent to pmac.

Definition at line 106 of file pgpmac.h.

#### 6.21.2.7 `pthread_cond_t lspmac_motor_struct::cond`

used to signal when a motor is done moving

Definition at line 103 of file pgpmac.h.

#### 6.21.2.8 `Isredis_obj_t* lspmac_motor_struct::coord_num`

coordinate system this motor belongs to (0 if none)

Definition at line 128 of file pgpmac.h.



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

controlling mvariable as a string

Definition at line 120 of file pgpmac.h.

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

pmac commands to home motor

Definition at line 131 of file pgpmac.h.

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

Homing routine started.

Definition at line 109 of file pgpmac.h.

**6.21.2.12 Isredis\_obj\_t\* lspmac\_motor\_struct::inactive\_init**

pmac commands to inactivate the motor

Definition at line 134 of file pgpmac.h.

**6.21.2.13 double\* lspmac\_motor\_struct::lut**

lookup table (instead of u2c)

Definition at line 142 of file pgpmac.h.

**6.21.2.14 Isredis\_obj\_t\* lspmac\_motor\_struct::max\_accel**

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

Definition at line 126 of file pgpmac.h.

**6.21.2.15 Isredis\_obj\_t\* lspmac\_motor\_struct::max\_speed**

our maximum speed (cts/msec)

Definition at line 125 of file pgpmac.h.

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

set to 1 when motion has been verified to have started

Definition at line 107 of file pgpmac.h.

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

pmac motor number

Definition at line 127 of file pgpmac.h.

**6.21.2.18** `void(* lspmac_motor_struct::moveAbs)(struct lspmac_motor_struct *, double)`

function to move the motor

Definition at line 141 of file pgpmac.h.

**6.21.2.19** `pthread_mutex_t lspmac_motor_struct::mutex`

coordinate waiting for motor to be done

Definition at line 102 of file pgpmac.h.

**6.21.2.20** `char* lspmac_motor_struct::name`

Name of motor as referred by ls database kvs table.

Definition at line 121 of file pgpmac.h.

**6.21.2.21** `int lspmac_motor_struct::nlut`

length of lut

Definition at line 143 of file pgpmac.h.

**6.21.2.22** `int lspmac_motor_struct::not_done`

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

Definition at line 104 of file pgpmac.h.

**6.21.2.23** `double lspmac_motor_struct::position`

scaled position

Definition at line 113 of file pgpmac.h.

**6.21.2.24** `pmac_cmd_queue_t* lspmac_motor_struct::pq`

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

Definition at line 108 of file pgpmac.h.

**6.21.2.25** `lsredis_obj_t* lspmac_motor_struct::printf_fmt`

printf format

Definition at line 123 of file pgpmac.h.

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

method to read the motor status and position

Definition at line 105 of file pgpmac.h.

**6.21.2.27 int lspmac\_motor\_struct::read\_mask**

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

Definition at line 140 of file pgpmac.h.

**6.21.2.28 int\* lspmac\_motor\_struct::read\_ptr**

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

Definition at line 139 of file pgpmac.h.

**6.21.2.29 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 124 of file pgpmac.h.

**6.21.2.30 Isredis\_obj\_t\* lspmac\_motor\_struct::redis\_position**

how we report our position to the world

Definition at line 135 of file pgpmac.h.

**6.21.2.31 double lspmac\_motor\_struct::reported\_position**

previous position reported to the database

Definition at line 114 of file pgpmac.h.

**6.21.2.32 int lspmac\_motor\_struct::requested\_pos\_cnts**

requested position

Definition at line 110 of file pgpmac.h.

**6.21.2.33 double lspmac\_motor\_struct::requested\_position**

The position as requested by the user.

Definition at line 115 of file pgpmac.h.

**6.21.2.34 int lspmac\_motor\_struct::status1**

local copy of status1

Definition at line 117 of file pgpmac.h.

**6.21.2.35 int\* lspmac\_motor\_struct::status1\_p**

First 24 bit PMAC motor status word.

Definition at line 116 of file pgpmac.h.

**6.21.2.36 int lspmac\_motor\_struct::status2**

local copy of status2

Definition at line 119 of file pgpmac.h.

**6.21.2.37 int\* lspmac\_motor\_struct::status2\_p**

Second 24 bit PMAC motor status word.

Definition at line 118 of file pgpmac.h.

**6.21.2.38 Isredis\_obj\_t\* lspmac\_motor\_struct::status\_str**

A talky version of the status.

Definition at line 136 of file pgpmac.h.

**6.21.2.39 Isredis\_obj\_t\* lspmac\_motor\_struct::u2c**

conversion from counts to units: 0.0 means not loaded yet

Definition at line 137 of file pgpmac.h.

**6.21.2.40 Isredis\_obj\_t\* lspmac\_motor\_struct::unit**

string to use as the units

Definition at line 122 of file pgpmac.h.

**6.21.2.41 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 129 of file pgpmac.h.

**6.21.2.42 WINDOW\* lspmac\_motor\_struct::win**

our ncurses window

Definition at line 144 of file pgpmac.h.

**6.21.2.43 char\* lspmac\_motor\_struct::write\_fmt**

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

Definition at line 138 of file pgpmac.h.

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

- [pgpmac.h](#)

## 6.22 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 [valid](#)  
*1 if we think the value is good, 0 otherwise*
- int [wait\\_for\\_me](#)  
*Number of times we need to see our publication before we start accepting new values.*
- 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*

### 6.22.1 Detailed Description

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

Definition at line 38 of file pgpmac.h.

### 6.22.2 Field Documentation

#### 6.22.2.1 char\*\* Isredis\_obj\_struct::avalue

our value as an array of strings

Definition at line 50 of file pgpmac.h.

#### 6.22.2.2 int Isredis\_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.

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

wait for a valid value

Definition at line 40 of file pgpmac.h.

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

just the first character of our value

Definition at line 52 of file pgpmac.h.

#### 6.22.2.5 double lsredis\_obj\_struct::dvalue

our value as a double

Definition at line 48 of file pgpmac.h.

#### 6.22.2.6 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.

#### 6.22.2.7 int lsredis\_obj\_struct::hits

number of times we've searched for this key

Definition at line 53 of file pgpmac.h.

#### 6.22.2.8 char\* lsredis\_obj\_struct::key

The redis key for this object.

Definition at line 44 of file pgpmac.h.

#### 6.22.2.9 long int lsredis\_obj\_struct::lvalue

our value as a long

Definition at line 49 of file pgpmac.h.

#### 6.22.2.10 pthread\_mutex\_t lsredis\_obj\_struct::mutex

Don't let anyone use an old value.

Definition at line 39 of file pgpmac.h.

#### 6.22.2.11 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.

## 6.22.2.12 char lsredis\_obj\_struct::valid

1 if we think the value is good, 0 otherwise

Definition at line 42 of file pgpmac.h.

## 6.22.2.13 char\* lsredis\_obj\_struct::value

our value

Definition at line 47 of file pgpmac.h.

## 6.22.2.14 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.

## 6.22.2.15 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](#)

## 6.23 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*

### 6.23.1 Detailed Description

Everything we need to know about a timer.

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

### 6.23.2 Field Documentation

#### 6.23.2.1 long int `lstimer_list_struct::delay_nsecs`

nano seconds of delay

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

#### 6.23.2.2 long int `lstimer_list_struct::delay_secs`

number of seconds for a periodic delay

Definition at line 28 of file `lstimer.c`.

#### 6.23.2.3 char `lstimer_list_struct::event[LSEVENTS_EVENT_LENGTH]`

the event to send

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

#### 6.23.2.4 long int `lstimer_list_struct::init_nsecs`

our initialization time

Definition at line 33 of file `lstimer.c`.

#### 6.23.2.5 long int `lstimer_list_struct::init_secs`

our initialization time

Definition at line 32 of file `lstimer.c`.

#### 6.23.2.6 long int `lstimer_list_struct::last_nsecs`

the last time this timer was triggered

Definition at line 31 of file `lstimer.c`.

#### 6.23.2.7 long int `lstimer_list_struct::last_secs`

the last time this timer was triggered

Definition at line 30 of file `lstimer.c`.



#### 6.23.2.8 unsigned long int ltimer\_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 ltimer.c.

#### 6.23.2.9 long int ltimer\_list\_struct::next\_nsecs

nano seconds of next alarm

Definition at line 27 of file ltimer.c.

#### 6.23.2.10 long int ltimer\_list\_struct::next\_secs

epoch (seconds) of next alarm

Definition at line 26 of file ltimer.c.

#### 6.23.2.11 int ltimer\_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 ltimer.c.

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

- [ltimer.c](#)

## 6.24 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 [copy\\_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](#)

### 6.24.1 Detailed Description

The block of memory retrieved in a status request.

Definition at line 243 of file `lspmac.c`.

## 6.24.2 Field Documentation

### 6.24.2.1 int md2StatusStruct::acc11c\_1

Definition at line 310 of file lspmac.c.

### 6.24.2.2 int md2StatusStruct::acc11c\_2

Definition at line 311 of file lspmac.c.

### 6.24.2.3 int md2StatusStruct::acc11c\_3

Definition at line 312 of file lspmac.c.

### 6.24.2.4 int md2StatusStruct::acc11c\_5

Definition at line 313 of file lspmac.c.

### 6.24.2.5 int md2StatusStruct::acc11c\_6

Definition at line 314 of file lspmac.c.

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

Definition at line 294 of file lspmac.c.

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

Definition at line 260 of file lspmac.c.

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

Definition at line 277 of file lspmac.c.

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

Definition at line 295 of file lspmac.c.

### 6.24.2.10 int md2StatusStruct::aligny\_status\_1

Definition at line 261 of file lspmac.c.

### 6.24.2.11 int md2StatusStruct::aligny\_status\_2

Definition at line 278 of file lspmac.c.

### 6.24.2.12 int md2StatusStruct::alignz\_act\_pos

Definition at line 296 of file lspmac.c.

6.24.2.13 `int md2StatusStruct::alignz_status_1`

Definition at line 262 of file `lspmac.c`.

6.24.2.14 `int md2StatusStruct::alignz_status_2`

Definition at line 279 of file `lspmac.c`.

6.24.2.15 `int md2StatusStruct::analyzer_act_pos`

Definition at line 297 of file `lspmac.c`.

6.24.2.16 `int md2StatusStruct::analyzer_status_1`

Definition at line 263 of file `lspmac.c`.

6.24.2.17 `int md2StatusStruct::analyzer_status_2`

Definition at line 280 of file `lspmac.c`.

6.24.2.18 `int md2StatusStruct::aperturey_act_pos`

Definition at line 299 of file `lspmac.c`.

6.24.2.19 `int md2StatusStruct::aperturey_status_1`

Definition at line 265 of file `lspmac.c`.

6.24.2.20 `int md2StatusStruct::aperturey_status_2`

Definition at line 282 of file `lspmac.c`.

6.24.2.21 `int md2StatusStruct::aperturez_act_pos`

Definition at line 300 of file `lspmac.c`.

6.24.2.22 `int md2StatusStruct::aperturez_status_1`

Definition at line 266 of file `lspmac.c`.

6.24.2.23 `int md2StatusStruct::aperturez_status_2`

Definition at line 283 of file `lspmac.c`.

6.24.2.24 `int md2StatusStruct::back_dac`

Definition at line 316 of file `lspmac.c`.

6.24.2.25 int md2StatusStruct::copy\_act\_pos

Definition at line 301 of file lspmac.c.

6.24.2.26 int md2StatusStruct::copy\_status\_1

Definition at line 267 of file lspmac.c.

6.24.2.27 int md2StatusStruct::copy\_status\_2

Definition at line 284 of file lspmac.c.

6.24.2.28 int md2StatusStruct::capz\_act\_pos

Definition at line 302 of file lspmac.c.

6.24.2.29 int md2StatusStruct::capz\_status\_1

Definition at line 268 of file lspmac.c.

6.24.2.30 int md2StatusStruct::capz\_status\_2

Definition at line 285 of file lspmac.c.

6.24.2.31 int md2StatusStruct::centerx\_act\_pos

Definition at line 304 of file lspmac.c.

6.24.2.32 int md2StatusStruct::centerx\_status\_1

Definition at line 270 of file lspmac.c.

6.24.2.33 int md2StatusStruct::centerx\_status\_2

Definition at line 287 of file lspmac.c.

6.24.2.34 int md2StatusStruct::centery\_act\_pos

Definition at line 305 of file lspmac.c.

6.24.2.35 int md2StatusStruct::centery\_status\_1

Definition at line 271 of file lspmac.c.

6.24.2.36 int md2StatusStruct::centery\_status\_2

Definition at line 288 of file lspmac.c.

6.24.2.37 int md2StatusStruct::dummy1

Definition at line 258 of file lspmacc.c.

6.24.2.38 int md2StatusStruct::dummy2

Definition at line 275 of file lspmacc.c.

6.24.2.39 int md2StatusStruct::dummy3

Definition at line 292 of file lspmacc.c.

6.24.2.40 int md2StatusStruct::dummy4

Definition at line 319 of file lspmacc.c.

6.24.2.41 int md2StatusStruct::dummy5

Definition at line 320 of file lspmacc.c.

6.24.2.42 int md2StatusStruct::dummy6

Definition at line 321 of file lspmacc.c.

6.24.2.43 int md2StatusStruct::dummy7

Definition at line 322 of file lspmacc.c.

6.24.2.44 int md2StatusStruct::dummy8

Definition at line 323 of file lspmacc.c.

6.24.2.45 int md2StatusStruct::dummy9

Definition at line 324 of file lspmacc.c.

6.24.2.46 int md2StatusStruct::dummyA

Definition at line 325 of file lspmacc.c.

6.24.2.47 int md2StatusStruct::dummyB

Definition at line 326 of file lspmacc.c.

6.24.2.48 int md2StatusStruct::front\_dac

Definition at line 315 of file lspmacc.c.

6.24.2.49 int md2StatusStruct::fs\_has\_opened

Definition at line 330 of file lspmac.c.

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

Definition at line 331 of file lspmac.c.

6.24.2.51 int md2StatusStruct::fs\_is\_open

Definition at line 328 of file lspmac.c.

6.24.2.52 int md2StatusStruct::kappa\_act\_pos

Definition at line 306 of file lspmac.c.

6.24.2.53 int md2StatusStruct::kappa\_status\_1

Definition at line 272 of file lspmac.c.

6.24.2.54 int md2StatusStruct::kappa\_status\_2

Definition at line 289 of file lspmac.c.

6.24.2.55 int md2StatusStruct::moving\_flags

Definition at line 334 of file lspmac.c.

6.24.2.56 int md2StatusStruct::number\_passes

Definition at line 332 of file lspmac.c.

6.24.2.57 int md2StatusStruct::omega\_act\_pos

Definition at line 293 of file lspmac.c.

6.24.2.58 int md2StatusStruct::omega\_status\_1

Definition at line 259 of file lspmac.c.

6.24.2.59 int md2StatusStruct::omega\_status\_2

Definition at line 276 of file lspmac.c.

6.24.2.60 int md2StatusStruct::phi\_act\_pos

Definition at line 307 of file lspmac.c.

#### 6.24.2.61 int md2StatusStruct::phi\_status\_1

Definition at line 273 of file lspmac.c.

#### 6.24.2.62 int md2StatusStruct::phi\_status\_2

Definition at line 290 of file lspmac.c.

#### 6.24.2.63 int md2StatusStruct::phiscan

Definition at line 329 of file lspmac.c.

#### 6.24.2.64 int md2StatusStruct::scint\_act\_pos

Definition at line 303 of file lspmac.c.

#### 6.24.2.65 int md2StatusStruct::scint\_piezo

Definition at line 317 of file lspmac.c.

#### 6.24.2.66 int md2StatusStruct::scint\_status\_1

Definition at line 269 of file lspmac.c.

#### 6.24.2.67 int md2StatusStruct::scint\_status\_2

Definition at line 286 of file lspmac.c.

#### 6.24.2.68 int md2StatusStruct::zoom\_act\_pos

Definition at line 298 of file lspmac.c.

#### 6.24.2.69 int md2StatusStruct::zoom\_status\_1

Definition at line 264 of file lspmac.c.

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

Definition at line 281 of file lspmac.c.

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

- [lspmac.c](#)

## 6.25 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.*

### 6.25.1 Detailed Description

PMAC ethernet packet definition.

Taken directly from the Delta Tau documentation.

Definition at line 73 of file pgpmac.h.

### 6.25.2 Field Documentation

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

The data buffer, if required.

Definition at line 79 of file pgpmac.h.

#### 6.25.2.2 unsigned char tagEthernetCmd::Request

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

Definition at line 75 of file pgpmac.h.

#### 6.25.2.3 unsigned char tagEthernetCmd::RequestType

VR\_UPLOAD or VR\_DOWNLOAD.

Definition at line 74 of file pgpmac.h.

#### 6.25.2.4 unsigned short tagEthernetCmd::wIndex

Command parameter 2.

Definition at line 77 of file pgpmac.h.

#### 6.25.2.5 unsigned short tagEthernetCmd::wLength

Number of bytes in bData.

Definition at line 78 of file pgpmac.h.

#### 6.25.2.6 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.jp](#) `iniParser("microdiff_hard.ini")`

### 7.2 kvredis.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <hiredis/hiredis.h>
#include <hiredis/async.h>
#include <poll.h>
#include <postgresql/libpq-fe.h>
#include <string.h>
```

#### Data Structures

- struct [lspgQueryQueueStruct](#)

*Store each query along with it's callback function.*

#### Macros

- `#define` [LS\\_PG\\_QUERY\\_QUEUE\\_LENGTH](#) 512

- `#define LS_PG_QUERY_STRING_LENGTH 512`
- `#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`

## Typedefs

- typedef struct `lspgQueryQueueStruct` `lspg_query_queue_t`

*Store each query along with it's callback function.*

## Functions

- void `redisDisconnectCB` (const `redisAsyncContext` \*ac, int status)
- void `debugCB` (`redisAsyncContext` \*ac, void \*reply, void \*privdata)
- void `addRead` (void \*data)
- void `delRead` (void \*data)
- void `addWrite` (void \*data)
- void `delWrite` (void \*data)
- void `cleanup` (void \*data)
- void `lspg_allkvs_cb` (`lspg_query_queue_t` \*qqp, `PGresult` \*pgr)
- PQnoticeProcessor `lspg_notice_processor` (void \*arg, const char \*msg)
- `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.*
- void `lspg_receive` ()  
*Receive a result of a query.*
- void `lspg_pg_connect` ()  
*Connect to the pg server.*
- void `lspg_flush` ()  
*Flush psql output buffer (ie, send the query)*
- 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_send_next_query` ()  
*send the next queued query to the DB server*
- void `lspg_pg_service` (struct pollfd \*evt)  
*I/O control to/from the postgresql server.*
- void `fd_service` (struct pollfd \*evt)
- `main` ()

## Variables

- static redisAsyncContext \* [subac](#)
- static redisAsyncContext \* [cmdac](#)
- 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 int [kvseq](#) = 0  
*used to synchronize pg.kvs and redis*
- 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.*
- static struct pollfd [lspgfd](#)  
*our poll info*
- static struct pollfd [subfd](#)  
*poll info for redis subscribe channel*
- static struct pollfd [cmdfd](#)  
*poll info for redis command channel*

### 7.2.1 Macro Definition Documentation

#### 7.2.1.1 #define LS\_PG\_QUERY\_QUEUE\_LENGTH 512

Definition at line 12 of file kvredis.c.

#### 7.2.1.2 #define LS\_PG\_QUERY\_STRING\_LENGTH 512

Definition at line 13 of file kvredis.c.

#### 7.2.1.3 #define LS\_PG\_STATE\_IDLE 1

Definition at line 19 of file kvredis.c.

#### 7.2.1.4 #define LS\_PG\_STATE\_INIT -4

Definition at line 15 of file kvredis.c.

#### 7.2.1.5 `#define LS_PG_STATE_INIT_POLL -3`

Definition at line 16 of file kvredis.c.

#### 7.2.1.6 `#define LS_PG_STATE_RECV 4`

Definition at line 22 of file kvredis.c.

#### 7.2.1.7 `#define LS_PG_STATE_RESET -2`

Definition at line 17 of file kvredis.c.

#### 7.2.1.8 `#define LS_PG_STATE_RESET_POLL -1`

Definition at line 18 of file kvredis.c.

#### 7.2.1.9 `#define LS_PG_STATE_SEND 2`

Definition at line 20 of file kvredis.c.

#### 7.2.1.10 `#define LS_PG_STATE_SEND_FLUSH 3`

Definition at line 21 of file kvredis.c.

### 7.2.2 Typedef Documentation

#### 7.2.2.1 `typedef struct lspgQueryQueueStruct lspg_query_queue_t`

Store each query along with it's callback function.

All calls are asynchronous

### 7.2.3 Function Documentation

#### 7.2.3.1 `void addRead ( void * data )`

Definition at line 111 of file kvredis.c.

```

    {
        struct pollfd *pfd;
        pfd = (struct pollfd *)data;
        pfd->events |= POLLIN;
    }

```

#### 7.2.3.2 `void addWrite ( void * data )`

Definition at line 121 of file kvredis.c.

```

    {
        struct pollfd *pfd;
        pfd = (struct pollfd *)data;
        pfd->events |= POLLOUT;
    }

```

**7.2.3.3 void cleanup ( void \* *data* )**

Definition at line 131 of file kvredis.c.

```

    {
        struct pollfd *pfd;
        pfd = (struct pollfd *)data;
        pfd->events &= ~(POLLOUT | POLLIN);
    }

```

**7.2.3.4 void debugCB ( redisAsyncContext \* *ac*, void \* *reply*, void \* *privdata* )**

Definition at line 63 of file kvredis.c.

```

    {
        static int indentlevel = 0;
        redisReply *r;
        int i;

        r = (redisReply *)reply;

        if( r == NULL ) {
            printf( "Null reply. Odd\n");
            return;
        }

        switch( r->type) {
        case REDIS_REPLY_STATUS:
            printf( "%sSTATUS: %s\n", indentlevel*4, "", r->str);
            break;

        case REDIS_REPLY_ERROR:
            printf( "%sERROR: %s\n", indentlevel*4, "", r->str);
            break;

        case REDIS_REPLY_INTEGER:
            printf( "%sInteger: %lld\n", indentlevel*4, "", r->integer);
            break;

        case REDIS_REPLY_NIL:
            printf( "%s(nil)\n", indentlevel*4, "");
            break;

        case REDIS_REPLY_STRING:
            printf( "%sSTRING: %s\n", indentlevel*4, "", r->str);
            break;

        case REDIS_REPLY_ARRAY:
            printf( "%sARRAY of %d elements\n", indentlevel*4, "", (int)r->elements);
            indentlevel++;
            for( i=0; i<r->elements; i++) {
                debugCB( ac, r->element[i], NULL);
            }
            indentlevel--;
            break;

        default:
            printf( "%sUnknown type %d\n", indentlevel*4, "", r->type);
        }
    }
}

```

**7.2.3.5 void delRead ( void \* *data* )**

Definition at line 116 of file kvredis.c.

```

    {
        struct pollfd *pfd;
        pfd = (struct pollfd *)data;
        pfd->events &= ~POLLIN;
    }

```

### 7.2.3.6 void delWrite ( void \* data )

Definition at line 126 of file kvredis.c.

```

    {
    struct pollfd *pfd;
    pfd = (struct pollfd *)data;
    pfd->events &= ~POLLOUT;
    }

```

### 7.2.3.7 void fd\_service ( struct pollfd \* evt )

Definition at line 636 of file kvredis.c.

```

    {
    if( evt->fd == subac->c.fd) {
        if( evt->revents & POLLIN)
            redisAsyncHandleRead( subac);
        if( evt->revents & POLLOUT)
            redisAsyncHandleWrite( subac);
    }
    if( evt->fd == cmdac->c.fd) {
        if( evt->revents & POLLIN)
            redisAsyncHandleRead( cmdac);
        if( evt->revents & POLLOUT)
            redisAsyncHandleWrite( cmdac);
    }
    if( q && evt->fd == PQsocket( q))
        lspg_pg_service( evt);
    }

```

### 7.2.3.8 void lspg\_allkvs\_cb ( lspg\_query\_queue\_t \* qqp, PGresult \* pgr )

Definition at line 137 of file kvredis.c.

```

    {
    int kvname_col, kvvalue_col, kvseq_col, kvdbrtype_col;
    int i;
    int seq;
    char *argv[8];

    kvname_col = PQfnumber( pgr, "rname");
    kvvalue_col = PQfnumber( pgr, "rvalue");
    kvseq_col = PQfnumber( pgr, "rseq");
    kvdbrtype_col = PQfnumber( pgr, "rdbrtype");

    if( kvname_col == -1 || kvvalue_col == -1 || kvseq_col == -1 || kvdbrtype_col
        == -1) {
        fprintf( stderr, "lspg_allkvs_cb: bad column number(s)\n");
        return;
    }

    redisAsyncCommand( cmdac, NULL, NULL, "MULTI");
    for( i=0; i<PQntuples( pgr); i++) {
        seq = atoi( PQgetvalue( pgr, i, kvseq_col));
        kvseq = kvseq < seq ? seq : kvseq;

        argv[0] = "HMSET";
        argv[1] = PQgetvalue( pgr, i, kvname_col);
        argv[2] = "VALUE";
        argv[3] = PQgetvalue( pgr, i, kvvalue_col);
        argv[4] = "SEQ";
        argv[5] = PQgetvalue( pgr, i, kvseq_col);
        argv[6] = "DBRTYPE";
        argv[7] = PQgetvalue( pgr, i, kvdbrtype_col);
        redisAsyncCommandArgv( cmdac, NULL, NULL, 8, (const char **)argv, NULL
            );

        argv[0] = "PUBLISH";
        argv[1] = "REDIS_KV_CONNECTOR";
        argv[2] = PQgetvalue( pgr, i, kvname_col);
        redisAsyncCommandArgv( cmdac, NULL, NULL, 3, (const char **)argv, NULL
            );
    }
    }

```



```

redisAsyncCommand( cmdac, NULL, NULL, "SET redis.kvseq %d", kvseq);

redisAsyncCommand( cmdac, NULL, NULL, "EXEC");
}

```

### 7.2.3.9 void lspg\_flush ( )

Flush psql output buffer (ie, send the query)

Definition at line 412 of file kvredis.c.

```

{
    int err;

    err = PQflush( q);
    switch( err) {
    case -1:
        // an error occurred

        fprintf( stderr, "flush failed: %s\n", 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;
    }
}

```

### 7.2.3.10 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.

Definition at line 444 of file kvredis.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)
        lspg_pg_connect( lspgfd);

    if( ls_pg_state == LS_PG_STATE_IDLE &&
        lspg_query_queue_on != lspg_query_queue_off
    )
        ls_pg_state = LS_PG_STATE_SEND;

    switch( ls_pg_state) {
    case LS_PG_STATE_INIT_POLL:
        if( lspg_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;
}
}

```

#### 7.2.3.11 PQnoticeProcessor lspg\_notice\_processor ( void \* arg, const char \* msg )

Definition at line 182 of file kvredis.c.

```

                                                                    {
    fprintf( stderr, "lspg: %s", msg);
}

```

#### 7.2.3.12 void lspg\_pg\_connect ( )

Connect to the pg server.

Definition at line 325 of file kvredis.c.

```

{
PGresult *pgr;
int wait_interval = 1;
int connection_init = 0;
int i, 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 the 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) {
        fprintf( stderr, "Out of memory (lspg_pg_connect)");
        exit( -1);
    }

    err = PQstatus( q);
    if( err == CONNECTION_BAD) {
        fprintf( stderr, "Trouble connecting to database");

        gettimeofday( &lspg_time_sent, NULL);
        return;
    }
}

```

```

    }
    err = PQsetnonblocking( q, 1);
    if( err != 0) {
        fprintf( stderr, "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);

        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) {
        ls_pg_state = LS_PG_STATE_IDLE;
    }
    break;
}
}

```

### 7.2.3.13 void lspg\_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 543 of file kvredis.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) {
                lspg_connectPoll_response = PQconnectPoll( q);
                if( lspg_connectPoll_response ==
                    PGRES_POLLING_FAILED) {

```

```

        ls_pg_state = LS_PG_STATE_RESET;
    }
    return;
}

if( ls_pg_state == LS_PG_STATE_RESET_POLL)
{
    lspg_resetPoll_response = PQresetPoll( q);
    if( lspg_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) {
        fprintf( stderr, "consume input failed: %s", PQerrorMessage( q));
        ls_pg_state = LS_PG_STATE_RESET;
        return;
    }
}

if( ls_pg_state == LS_PG_STATE_RECV) {
    lspg_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;

        lspg_query_push( lspg_allkvs_cb, "SELECT *
        FROM px.redis_kv_update(%d)", kvseq);

        PQfreemem( pgn);
    }
}

if( evt->revents & POLLOUT) {

    if( ls_pg_state == LS_PG_STATE_INIT_POLL) {
        lspg_connectPoll_response = PQconnectPoll( q);
        if( lspg_connectPoll_response ==
        PGRES_POLLING_FAILED) {
            ls_pg_state = LS_PG_STATE_RESET;
        }
        return;
    }

    if( ls_pg_state == LS_PG_STATE_RESET_POLL)
    {
        lspg_resetPoll_response = PQresetPoll( q);
        if( lspg_resetPoll_response ==
        PGRES_POLLING_FAILED) {
            ls_pg_state = LS_PG_STATE_RESET;
        }
        return;
    }

    if( ls_pg_state == LS_PG_STATE_SEND) {
        lspg_send_next_query();
    }

    if( ls_pg_state == LS_PG_STATE_SEND_FLUSH)
    {
        lspg_flush();
    }
}
}

```

### 7.2.3.14 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 191 of file kvredis.c.

```

{
    lspg_query_queue_t *rtn;

    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]);
    }
    return rtn;
}

```

### 7.2.3.15 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 234 of file kvredis.c.

```

{
    int idx;
    va_list arg_ptr;

    //
    // Pause the thread while we service the queue
    //
    if( (lspg_query_queue_on + 1) % LS_PG_QUERY_QUEUE_LENGTH
        == lspg_query_queue_off % LS_PG_QUERY_QUEUE_LENGTH
    ) {
        fprintf( stderr, "lspg_query_push: queue is full. Ignoring query \"%s\"\n"
        , fmt);
        return;
    }

    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++;
};

```

### 7.2.3.16 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 211 of file kvredis.c.

```

    {
        if( lspg_query_queue_reply != lspg_query_queue_on
        )
            lspg_query_queue_reply++;
    }

```

### 7.2.3.17 `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 221 of file kvredis.c.

```

    {
        lspg_query_queue_t *rtn;

        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]);

        return rtn;
    }

```

### 7.2.3.18 `void` `lspg_receive ( )`

Receive a result of a query.

Definition at line 266 of file kvredis.c.

```

    {
        PGresult *pgr;
        lspg_query_queue_t *qqp;
        int err;

        err = PQconsumeInput( q );
        if( err != 1 ) {
            fprintf( stderr, "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 ) {
                lspg_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 = lspg_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) {
        fprintf( stderr, "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 look 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);
}
}
}

```

### 7.2.3.19 void lspg\_send\_next\_query ( )

send the next queued query to the DB server

Definition at line 496 of file kvredis.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
        //
        fprintf( stderr, "Popped empty query string. Probably bad things are going
            on.\n");

        lspg_query_reply_next();
        ls_pg_state = LS_PG_STATE_IDLE;
    }
    else {
        err = PQsendQuery( q, qqp->qs);
        if( err == 0) {
            fprintf( stderr, "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;
        }
    }
}
}

```

### 7.2.3.20 main ( )

Definition at line 655 of file kvredis.c.

```

    {
static struct pollfd fda[3];
static int nfda = 0;
int pollrtn;
int poll_timeout_ms;
int i;

subac = redisAsyncConnect("127.0.0.1", 6379);
if( subac->err) {
    fprintf( stderr, "Error: %s\n", subac->errstr);
    exit( -1);
}

cmdac = redisAsyncConnect("127.0.0.1", 6379);
if( cmdac->err) {
    fprintf( stderr, "Error: %s\n", cmdac->errstr);
    exit( -1);
}

if( redisAsyncSetDisconnectCallback( subac, redisDisconnectCB
    ) == REDIS_ERR) {
    fprintf( stderr, "Error: could not set disconnect callback\n");
    exit( -1);
}

if( redisAsyncSetDisconnectCallback( cmdac, redisDisconnectCB
    ) == REDIS_ERR) {
    fprintf( stderr, "Error: could not set disconnect callback\n");
    exit( -1);
}

// Set up redis events
//
subfd.fd      = subac->c.fd;
subfd.events  = 0;
subac->ev.data = &subfd;
subac->ev.addRead  = addRead;
subac->ev.delRead  = delRead;
subac->ev.addWrite = addWrite;
subac->ev.delWrite = delWrite;
subac->ev.cleanup  = cleanup;

cmdfd.fd      = cmdac->c.fd;
cmdfd.events  = 0;
cmdac->ev.data = &cmdfd;
cmdac->ev.addRead  = addRead;
cmdac->ev.delRead  = delRead;
cmdac->ev.addWrite = addWrite;
cmdac->ev.delWrite = delWrite;
cmdac->ev.cleanup  = cleanup;

lspgfd.fd = -1;

if( redisAsyncCommand( cmdac, NULL, NULL, "KEYS *" ) == REDIS_ERR) {
    fprintf( stderr, "Error sending KEYS command\n");
    exit( -1);
}

if( redisAsyncCommand( subac, debugCB, NULL, "PSUBSCRIBE MD2* UI*"
    ) == REDIS_ERR) {
    fprintf( stderr, "Error sending PSUBSCRIBE command\n");
    exit( -1);
}

lspg_query_push( lspg_allkvs_cb, "SELECT * FROM
    px.redis_kv_init()");
lspg_query_push( NULL, "LISTEN REDIS_KV_CONNECTOR");

while( 1) {
    nfda = 0;
    if( subfd.fd != -1) {
        fda[nfda].fd      = subfd.fd;
        fda[nfda].events  = subfd.events;
        fda[nfda].revents = 0;

        nfda++;
    }
    if( cmdfd.fd != -1) {
        fda[nfda].fd      = cmdfd.fd;
        fda[nfda].events  = cmdfd.events;
        fda[nfda].revents = 0;

        nfda++;
    }
    poll_timeout_ms = -1;

```



```

lspg_next_state();

if( lspgfd.fd == -1) {
    //
    // Here a connection to the database is not established.
    // Periodicaly try again. Should possibly arrange to reconnect
    // to signalfd but that's unlikely to be nessesary.
    //
    poll_timeout_ms = 10000;
} else {
    //
    // Arrange to peacefully do nothing until either the pg server sends us
    // something
    // or someone pushes something onto our queue
    //
    fda[nfda].fd      = lspgfd.fd;
    fda[nfda].events  = lspgfd.events;
    fda[nfda].revents = 0;
    nfda++;
    poll_timeout_ms = -1;
}

pollrtn = poll( fda, nfda, poll_timeout_ms);

for( i=0; i<nfda; i++) {
    if( fda[i].revents) {
        fd_service( &{fda[i]});
    }
}
}
}

```

### 7.2.3.21 void redisDisconnectCB ( const redisAsyncContext \* ac, int status )

Definition at line 54 of file kvredis.c.

```

{
if( status == REDIS_OK) {
    printf( "OK, that was fun.\n");
    exit( 0);
}
fprintf( stderr, "Oops, Disconnected with status %d\n", status);
exit( -1);
}

```

## 7.2.4 Variable Documentation

### 7.2.4.1 redisAsyncContext \* cmdac [static]

Definition at line 9 of file kvredis.c.

### 7.2.4.2 struct pollfd cmdfd [static]

poll info for redis command channel

Definition at line 50 of file kvredis.c.

### 7.2.4.3 int kvseq = 0 [static]

used to synchronize pg.kvs and redis

Definition at line 26 of file kvredis.c.

### 7.2.4.4 int ls\_pg\_state = LS\_PG\_STATE\_INIT [static]

State of the lspg state machine.

Definition at line 24 of file kvredis.c.

#### 7.2.4.5 `PostgresPollingStatusType lspg_connectPoll_response` `[static]`

Used to determine state while connecting.

Definition at line 46 of file kvredis.c.

#### 7.2.4.6 `lspg_query_queue_t lspg_query_queue[LS_PG_QUERY_QUEUE_LENGTH]` `[static]`

Our query queue.

Definition at line 37 of file kvredis.c.

#### 7.2.4.7 `unsigned int lspg_query_queue_off = 0` `[static]`

The last item still being used (on == off means nothing in queue)

Definition at line 39 of file kvredis.c.

#### 7.2.4.8 `unsigned int lspg_query_queue_on = 0` `[static]`

Next position to add something to the queue.

Definition at line 38 of file kvredis.c.

#### 7.2.4.9 `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 40 of file kvredis.c.

#### 7.2.4.10 `PostgresPollingStatusType lspg_resetPoll_response` `[static]`

Used to determine state while reconnecting.

Definition at line 47 of file kvredis.c.

#### 7.2.4.11 `struct pollfd lspgfd` `[static]`

our poll info

Definition at line 48 of file kvredis.c.

#### 7.2.4.12 `struct timeval lspg_time_sent now` `[static]`

used to ensure we do not inundate the db server with connection requests

Definition at line 25 of file kvredis.c.

**7.2.4.13** PGconn\* q = NULL [static]

Database connector.

Definition at line 45 of file kvredis.c.

**7.2.4.14** redisAsyncContext\* subac [static]

Definition at line 9 of file kvredis.c.

**7.2.4.15** struct pollfd subfd [static]

poll info for redis subscribe channel

Definition at line 49 of file kvredis.c.

## 7.3 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.*

### Macros

- #define [LSEVENTS\\_QUEUE\\_LENGTH](#) 512

### Typedefs

- typedef struct  
[lsevents\\_queue\\_struct](#) lsevents\_queue\_t  
*Storage definition for the events.*
- typedef struct  
[lsevents\\_listener\\_struct](#) lsevents\_listener\_t  
*Linked list of event listeners.*

### Functions

- void [lsevents\\_send\\_event](#) (char \*fmt,...)  
*Call the callback routines for the given event.*
- void [lsevents\\_add\\_listener](#) (char \*event, void(\*cb)(char \*))  
*Add a callback routine to listen for a specific event.*
- void [lsevents\\_remove\\_listener](#) (char \*event, void(\*cb)(char \*))  
*Remove a listener previously added with lsevents\_add\_listener.*

- void \* [lsevents\\_worker](#) (void \*dummy)  
*Our worker.*
- void [lsevents\\_init](#) ()  
*Initialize this module.*
- void [lsevents\\_run](#) ()  
*Start up the thread and get out of the way.*

## Variables

- static [lsevents\\_queue\\_t](#) [lsevents\\_queue](#) [[LSEVENTS\\_QUEUE\\_LENGTH](#)]  
*simple list of events*
- static unsigned int [lsevents\\_queue\\_on](#) = 0  
*next queue location to write*
- static unsigned int [lsevents\\_queue\\_off](#) = 0  
*next queue location to read*
- static [lsevents\\_listener\\_t](#) \* [lsevents\\_listeners\\_p](#) = NULL  
*Pointer to the first item in the link list of listeners.*
- 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*

### 7.3.1 Detailed Description

event subsystem for inter-pgpmac communication

#### Date

2012

#### Author

Keith Brister

#### Copyright

All Rights Reserved

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

### 7.3.2 Macro Definition Documentation

#### 7.3.2.1 #define LSEVENTS\_QUEUE\_LENGTH 512

Definition at line 10 of file [lsevents.c](#).

### 7.3.3 Typedef Documentation

#### 7.3.3.1 typedef struct lsevents\_listener\_struct lsevents\_listener\_t

Linked list of event listeners.

#### 7.3.3.2 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.

### 7.3.4 Function Documentation

#### 7.3.4.1 void lsevents\_add\_listener ( char \* event, void(\*)(char \*) cb )

Add a callback routine to listen for a specific event.

##### Parameters

<i>event</i>	the name of the event to listen for
<i>cb</i>	the routine to call

Definition at line 75 of file lsevents.c.

```

lsevents_listener_t *new;
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, event, 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( event);
new->cb = cb;

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

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

```

### 7.3.4.2 void lsevents\_init ( )

Initialize this module.

Definition at line 214 of file lsevents.c.

```

{
    pthread_mutex_init( &lsevents_queue_mutex, NULL);
    pthread_cond_init( &lsevents_queue_cond, NULL);
    pthread_mutex_init( &lsevents_listener_mutex, NULL);
}

```

### 7.3.4.3 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
<i>cb</i>	The callback routine to remove

Definition at line 120 of file lsevents.c.

```

{
    lsevents_listener_t *last, *current;

    //
    // 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;
        }
    }
    pthread_mutex_unlock( &lsevents_listener_mutex);

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

```

### 7.3.4.4 void lsevents\_run ( )

Start up the thread and get out of the way.

Definition at line 222 of file lsevents.c.

```

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

```

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

Call the callback routines for the given event.

**Parameters**

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

Definition at line 45 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);

    lslogging_log_message( "lsevents_send_event: %s", event)
        ;

    // 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);
}

```

**7.3.4.6 void\* lsevents\_worker ( void \* *dummy* )**

Our worker.

**Parameters**

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

Definition at line 155 of file lsevents.c.

```

{

    // char *event;
    lsevents_queue_t *ep;
    lsevents_listener_t *p;

    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);

        //
        // copy event string since the value in the queue may change when
        // we unlock the mutex
        //
        ep = &(lsevents_queue[(lsevents_queue_off++)
            % LSEVENTS_QUEUE_LENGTH]);
    }
}

```

```

//
// 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);

//
// Find the callbacks and, well, call them back
//
// TODO:
//
// Yes, this is O(N).
//
// Plan to make this O(1):
//   track actual event names from send_event
//   match listeners for new event names
//   store matches in hash table
//
// That makes send_event for new events O(N)
// but O(1) otherwise, O(N) for add_listener, and O(1) here.
//
pthread_mutex_lock( &lsevents_listener_mutex);
for( p = lsevents_listeners_p; p != NULL; p = p->next
) {
    if( regexec( &p->re, ep->evp, 0, NULL, 0) == 0) {
        p->cb( ep->evp);
    }
}
free( ep->evp);

pthread_mutex_unlock( &lsevents_listener_mutex);
}
return NULL;
}

```

### 7.3.5 Variable Documentation

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

mutex to protect the listener linked list

Definition at line 37 of file lsevents.c.

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

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

Definition at line 34 of file lsevents.c.

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

simple list of events

Definition at line 21 of file lsevents.c.

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

condition to pause the queue if needed

Definition at line 39 of file lsevents.c.

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

mutex to protect the event queue

Definition at line 38 of file lsevents.c.



7.3.5.6 `unsigned int lsevents_queue_off = 0` `[static]`

next queue location to read

Definition at line 23 of file lsevents.c.

7.3.5.7 `unsigned int lsevents_queue_on = 0` `[static]`

next queue location to write

Definition at line 22 of file lsevents.c.

7.3.5.8 `pthread_t lsevents_thread` `[static]`

thread to run the event queue

Definition at line 36 of file lsevents.c.

## 7.4 Islogging.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\\_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*

### 7.4.1 Detailed Description

Logs messages to a file.

#### Date

2012

#### Author

Keith Brister

#### Copyright

All Rights Reserved

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

### 7.4.2 Macro Definition Documentation

#### 7.4.2.1 #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](#).

#### 7.4.2.2 #define LSLOGGING\_MSG\_LENGTH 2048

Fixed maximum length messages to keep some form of sanity.

Definition at line 20 of file [lslogging.c](#).

#### 7.4.2.3 #define LSLOGGING\_QUEUE\_LENGTH 8192

Modest length queue.

Definition at line 30 of file Islogging.c.

### 7.4.3 Typedef Documentation

#### 7.4.3.1 typedef struct Islogging\_queue\_struct Islogging\_queue\_t

Our log object: time and message.

### 7.4.4 Function Documentation

#### 7.4.4.1 void Islogging\_init ( )

Initialize the Islogging objects.

Definition at line 37 of file Islogging.c.

```

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

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

```

#### 7.4.4.2 void Islogging\_log\_message ( char \* fmt, ... )

The routine everyone will be talking about.

##### Parameters

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

Definition at line 48 of file Islogging.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);
}

```

#### 7.4.4.3 void lslogging\_run ( )

Start up the worker thread.

Definition at line 105 of file lslogging.c.

```

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

```

#### 7.4.4.4 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 76 of file lslogging.c.

```

    {

        struct tm coarsetime;
        char tstr[64];
        unsigned int msec;
        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;
            msec = lslogging_queue[off].ltime.tv_nsec / 1000;
            fprintf( lslogging_file, "%s.%06u %s\n", tstr, msec,
                lslogging_queue[off].lmsg);
            fflush( lslogging_file);
        }
    }

```

### 7.4.5 Variable Documentation

#### 7.4.5.1 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.

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

our log file object

Definition at line 17 of file lslogging.c.

#### 7.4.5.3 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.

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

next location to remove from the queue

Definition at line 34 of file lslogging.c.

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

next location to add to the queue

Definition at line 33 of file lslogging.c.

#### 7.4.5.6 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.

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

our thread

Definition at line 10 of file lslogging.c.

## 7.5 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\\_diffractionmeter\\_struct](#)  
*Object used to impliment 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_diffractionmeter_struct` `lspg_lock_diffractionmeter_t`  
*Object used to impliment 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_starttransfer_init ()`
- void `lspg_starttransfer_cb (lspg_query_queue_t *qqp, PGresult *pgr)`
- 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_wait ()`
- void `lspg_starttransfer_done ()`
- int `lspg_starttransfer_all (int *err, unsigned int nextsample, int sampledetected, 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_nextsample_cb (lspg_query_queue_t *qqp, PGresult *pgr)`

- Next Sample.*
  - void `lspg_nextsample_init ()`  
*Initialize the nextsample variable, mutex, and condition.*
  - void `lspg_nextsample_call ()`  
*Queue up a nextsample query.*
  - void `lspg_nextsample_wait ()`  
*Wait for the nextsample query to get processed.*
  - void `lspg_nextsample_done ()`  
*Called when the next shot query has been processed.*
  - unsigned int `lspg_nextsample_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 ()`  
*wait for the air rights request to return*
  - void `lspg_demandairrights_all ()`  
*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_diffractionmeter_init ()`  
*initialize the diffractionmeter locking object*
  - void `lspg_lock_diffractionmeter_cb (lspg_query_queue_t *qqp, PGresult *pgr)`  
*Callback routine for a lock diffractionmeter query.*
  - void `lspg_lock_diffractionmeter_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 ()`
- Convenience 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 convenience 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)`
- Convenience 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_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)
- 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 [lspmac\\_sample\\_detector\\_cb](#) (char \*event)  
*log magnet state*
- void [lspg\\_init](#) ()  
*Initialize the lspg module.*
- void [lspg\\_run](#) ()  
*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\\_nextsample\\_t](#) [lspg\\_nextsample](#)  
*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\\_diffractionmeter\\_t](#) [lspg\\_lock\\_diffractionmeter](#)
- static [lspg\\_lock\\_detector\\_t](#) [lspg\\_lock\\_detector](#)
- static [lspg\\_seq\\_run\\_prep\\_t](#) [lspg\\_seq\\_run\\_prep](#)

### 7.5.1 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](#).

### 7.5.2 Macro Definition Documentation

#### 7.5.2.1 #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 51 of file [lspg.c](#).

#### 7.5.2.2 `#define LS_PG_STATE_IDLE 1`

Definition at line 34 of file lspg.c.

#### 7.5.2.3 `#define LS_PG_STATE_INIT -4`

Definition at line 30 of file lspg.c.

#### 7.5.2.4 `#define LS_PG_STATE_INIT_POLL -3`

Definition at line 31 of file lspg.c.

#### 7.5.2.5 `#define LS_PG_STATE_RECV 4`

Definition at line 37 of file lspg.c.

#### 7.5.2.6 `#define LS_PG_STATE_RESET -2`

Definition at line 32 of file lspg.c.

#### 7.5.2.7 `#define LS_PG_STATE_RESET_POLL -1`

Definition at line 33 of file lspg.c.

#### 7.5.2.8 `#define LS_PG_STATE_SEND 2`

Definition at line 35 of file lspg.c.

#### 7.5.2.9 `#define LS_PG_STATE_SEND_FLUSH 3`

Definition at line 36 of file lspg.c.

### 7.5.3 Typedef Documentation

#### 7.5.3.1 `typedef struct lspg_lock_detector_struct lspg_lock_detector_t`

lock detector object Implements detector lock for exposure control

#### 7.5.3.2 `typedef struct lspg_lock_diffractionmeter_struct lspg_lock_diffractionmeter_t`

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

#### 7.5.3.3 `typedef struct lspg_seq_run_prep_struct lspg_seq_run_prep_t`

Data collection running object.

#### 7.5.3.4 `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.

## 7.5.4 Function Documentation

### 7.5.4.1 char\*\* lspg\_array2ptrs ( char \* a )

returns a null terminated list of strings parsed from postgresql array

Definition at line 161 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);
}

```

#### 7.5.4.2 void lspg\_cmd\_cb ( lspg\_query\_queue\_t \* qqp, PGresult \* pgr )

Send strings directly to PMAC queue.

##### Parameters

in	<i>qqp</i>	Our query
in	<i>pgr</i>	Our result

Definition at line 1238 of file lspg.c.

```

    {
        //
        // Call back function assumes query results in zero or more commands to send
        // to the PMAC
        //
        int i;
        char *sp;

        for( i=0; i<PGntuples( pgr); i++) {
            sp = PQgetvalue( pgr, i, 0);
            if( sp != NULL && *sp != 0) {
                lspmac_SockSendDPLine( NULL, sp);
                // lspmac_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()");
            }
        }
    }
}

```

**7.5.4.3 void lspg\_demandairrights\_all ( )**

do nothing until we get airrights

Definition at line 556 of file lspg.c.

```

    {
        lspg_demandairrights_call();
        lspg_demandairrights_wait();
        // there is no "done" version
    }

```

**7.5.4.4 void lspg\_demandairrights\_call ( )**

call for airrights

Definition at line 538 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()");
    }

```

**7.5.4.5 void lspg\_demandairrights\_cb ( lspg\_query\_queue\_t \* qqp, PGresult \* pgr )**

handle the airrights response

Definition at line 529 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);
    }

```

**7.5.4.6 void lspg\_demandairrights\_init ( )**

initialize the demandairrights structure

Definition at line 521 of file lspg.c.

```

    {
        lspg_demandairrights.new_value_ready = 0;
        pthread_mutex_init( &lspg_demandairrights.mutex,
            NULL);
        pthread_cond_init( &lspg_demandairrights.cond, NULL);
    }

```

**7.5.4.7 void lspg\_demandairrights\_wait ( )**

wait for the air rights request to return

Definition at line 547 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);
    }

```

## 7.5.4.8 void lspg\_flush ( )

Flush psql output buffer (ie, send the query)

Definition at line 1268 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;
        }
    }
}

```

## 7.5.4.9 void lspg\_getcenter\_all ( )

Convenience function to complete synchronous getcenter query.

Definition at line 1201 of file lspg.c.

```

    {
        lspg_getcenter_call();
        lspg_getcenter_wait();
        lspg_getcenter_done();
    }
}

```

## 7.5.4.10 void lspg\_getcenter\_call ( )

Request a getcenter query.

Definition at line 1177 of file lspg.c.

```

    {
        pthread_mutex_lock( &lspg_getcenter.mutex);
        lspg_getcenter.new_value_ready = 0;
        pthread_mutex_unlock( &lspg_getcenter.mutex);

        lspg_query_push( lspg_getcenter_cb, "SELECT *
            FROM px.getcenter2() ");
    }
}

```

## 7.5.4.11 void lspg\_getcenter\_cb ( lspg\_query\_queue\_t \* qq, PGresult \* pgr )

Retrieve the data to center the crystal.

Definition at line 1112 of file lspg.c.

```

static int
    zoom_c, dcx_c, dcy_c, dax_c, day_c, daz_c;

pthread_mutex_lock( &(lspg_getcenter.mutex));

lspg_getcenter.no_rows_returned = PQntuples(
    pgr) <= 0;
if( lspg_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.
    //
    lspg_getcenter.new_value_ready = 1;
    pthread_cond_signal( &(lspg_getcenter.cond));
    pthread_mutex_unlock( &(lspg_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));
}

```

#### 7.5.4.12 void lspg\_getcenter\_done ( )

Done with getcenter query.

Definition at line 1195 of file lspg.c.

```

{
    pthread_mutex_unlock( &(lspg_getcenter.mutex));
}

```

#### 7.5.4.13 void lspg\_getcenter\_init ( )

Initialize getcenter object.

Definition at line 1169 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);
    }

```

#### 7.5.4.14 void lspg\_getcenter\_wait ( )

Wait for a getcenter query to return.

Definition at line 1187 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));
    }

```

#### 7.5.4.15 void lspg\_getcurrentsampleid\_call ( )

Definition at line 367 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()");
    }

```

#### 7.5.4.16 void lspg\_getcurrentsampleid\_cb ( lspg\_query\_queue\_t \* qqp, PGresult \* pgr )

get currentsampleid

Definition at line 346 of file lspg.c.

```

    {
        pthread_mutex_lock( &lspg_getcurrentsampleid.mutex
        );

        lspg_nextsample.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);

        pthread_cond_signal( &lspg_getcurrentsampleid.cond
        );
        pthread_mutex_unlock( &lspg_getcurrentsampleid.mutex
        );
    }

```

**7.5.4.17 void lspg\_getcurrentsampleid\_init ( )**

Definition at line 338 of file lspg.c.

```

{
    lspg_getcurrentsampleid.new_value_ready
        = 0;
    pthread_mutex_init( &lspg_getcurrentsampleid.mutex
        , NULL);
    pthread_cond_init( &lspg_getcurrentsampleid.cond,
        NULL);
}

```

**7.5.4.18 unsigned int lspg\_getcurrentsampleid\_read ( )**

Definition at line 377 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;
}

```

**7.5.4.19 void lspg\_getcurrentsampleid\_wait\_for\_id ( unsigned int test )**

Definition at line 393 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
        );
}

```

**7.5.4.20 void lspg\_init ( )**

Initialize the lspg module.

Definition at line 1758 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_diffractionmeter_init();
    lspg_nextsample_init();
}

```

```

lspg_nextshot_init();
lspg_seq_run_prep_init();
lspg_starttransfer_init();
lspg_wait_for_detector_init();
lspg_waitcryo_init();
}

```

#### 7.5.4.21 void lspg\_lock\_detector\_all ( )

Detector lock convenience function.

Definition at line 1024 of file lspg.c.

```

{
    lspg_lock_detector_call();
    lspg_lock_detector_wait();
    lspg_lock_detector_done();
}

```

#### 7.5.4.22 void lspg\_lock\_detector\_call ( )

Request (demand) a detector lock.

Definition at line 1000 of file lspg.c.

```

{
    pthread_mutex_lock( &(lspg_lock_detector.mutex));
    lspg_lock_detector.new_value_ready = 0;
    pthread_mutex_unlock( &(lspg_lock_detector.mutex));

    lspg_query_push( lspg_lock_detector_cb, "
        SELECT px.lock_detector() ");
}

```

#### 7.5.4.23 void lspg\_lock\_detector\_cb ( lspg\_query\_queue\_t \* qqp, PGresult \* pgr )

Callback for when the detector lock has be grabbed.

Definition at line 991 of file lspg.c.

```

{
    pthread_mutex_lock( &(lspg_lock_detector.mutex));
    lspg_lock_detector.new_value_ready = 1;
    pthread_cond_signal( &(lspg_lock_detector.cond));
    pthread_mutex_unlock( &(lspg_lock_detector.mutex));
}

```

#### 7.5.4.24 void lspg\_lock\_detector\_done ( )

Finish waiting.

Definition at line 1018 of file lspg.c.

```

{
    pthread_mutex_unlock( &(lspg_lock_detector.mutex));
}

```

**7.5.4.25 void lspg\_lock\_detector\_init ( )**

Initialize detector lock object.

Definition at line 983 of file lspg.c.

```

{
    lspg_lock_detector.new_value_ready = 0;
    pthread_mutex_init( &(lspg_lock_detector.mutex), NULL)
    ;
    pthread_cond_init( &(lspg_lock_detector.cond), NULL);
}

```

**7.5.4.26 void lspg\_lock\_detector\_wait ( )**

Wait for the detector lock.

Definition at line 1010 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));
}

```

**7.5.4.27 void lspg\_lock\_diffractionmeter\_all ( )**

Convenience function that combines lock diffractionmeter calls.

Definition at line 965 of file lspg.c.

```

{
    lspg_lock_diffractionmeter_call();
    lspg_lock_diffractionmeter_wait();
    lspg_lock_diffractionmeter_all();
}

```

**7.5.4.28 void lspg\_lock\_diffractionmeter\_call ( )**

Request that the database grab the diffractionmeter lock.

Definition at line 941 of file lspg.c.

```

{
    pthread_mutex_lock( &(lspg_lock_diffractionmeter.mutex
    ));
    lspg_lock_diffractionmeter.new_value_ready
    = 0;
    pthread_mutex_unlock( &(lspg_lock_diffractionmeter.
    mutex));

    lspg_query_push( lspg_lock_diffractionmeter_cb
    , "SELECT px.lock_diffractionmeter()");
}

```

**7.5.4.29 void lspg\_lock\_diffractionmeter\_cb ( lspg\_query\_queue\_t \* qqp, PGresult \* pgr )**

Callback routine for a lock diffractionmeter query.

Definition at line 932 of file lspg.c.

```

pthread_mutex_lock( &(lspg_lock_diffractionmeter.mutex
));
lspg_lock_diffractionmeter.new_value_ready
= 1;
pthread_cond_signal( &(lspg_lock_diffractionmeter.cond
));
pthread_mutex_unlock( &(lspg_lock_diffractionmeter.
mutex));
}

```

#### 7.5.4.30 void lspg\_lock\_diffractionmeter\_done ( )

Finish up the lock diffractionmeter call.

Definition at line 959 of file lspg.c.

```

{
pthread_mutex_unlock( &(lspg_lock_diffractionmeter.
mutex));
}

```

#### 7.5.4.31 void lspg\_lock\_diffractionmeter\_init ( )

initialize the diffractionmeter locking object

Definition at line 924 of file lspg.c.

```

{
lspg_lock_diffractionmeter.new_value_ready
= 0;
pthread_mutex_init( &(lspg_lock_diffractionmeter.mutex
), NULL);
pthread_cond_init( &(lspg_lock_diffractionmeter.cond
), NULL);
}

```

#### 7.5.4.32 void lspg\_lock\_diffractionmeter\_wait ( )

Wait for the diffractionmeter lock.

Definition at line 951 of file lspg.c.

```

{
pthread_mutex_lock( &(lspg_lock_diffractionmeter.mutex
));
while( lspg_lock_diffractionmeter.new_value_ready
== 0)
pthread_cond_wait( &(lspg_lock_diffractionmeter.cond
), &(lspg_lock_diffractionmeter.mutex));
}

```

#### 7.5.4.33 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.

Definition at line 1623 of file lspg.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)
    lspg_pg_connect( lspgfd);

if( ls_pg_state == LS_PG_STATE_IDLE &&
    lspg_query_queue_on != lspg_query_queue_off
    )
    ls_pg_state = LS_PG_STATE_SEND;

switch( ls_pg_state) {
case LS_PG_STATE_INIT_POLL:
    if( lspg_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;
}
}

```

#### 7.5.4.34 void lspg\_nextaction.cb ( lspg\_query\_queue\_t \* qqp, PGresult \* pgr )

Queue the next MD2 instruction.

##### Parameters

in	<i>qqp</i>	The query that generated this result
in	<i>pgr</i>	The result

Definition at line 1210 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;

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( "MD2 command '%s' ignored.
                          Already running '%s'", action, md2cmds_cmd);
}
}

```

#### 7.5.4.35 unsigned int lspg\_nextsample\_all( int \* err )

Definition at line 468 of file lspg.c.

```

{
    unsigned int rtn;

    lspg_nextsample_call();
    lspg_nextsample_wait();

    if( lspg_nextsample.no_rows_returned) {
        rtn = 0;
        *err = 1;
    } else {
        if( lspg_nextsample.nextsample_isnull) {
            rtn = 0;
            *err = 1;
        } else {
            rtn = lspg_nextsample.nextsample;
            *err = 0;
        }
    }
    lspg_nextsample_done();

    return rtn;
}

```

#### 7.5.4.36 void lspg\_nextsample\_call( )

Queue up a nextsample query.

Definition at line 445 of file lspg.c.

```

{
    pthread_mutex_lock( &(lspg_nextsample.mutex));
    lspg_nextsample.new_value_ready = 0;
    pthread_mutex_unlock( &(lspg_nextsample.mutex));

    lspg_query_push( lspg_nextsample.cb, "SELECT
        nextsample FROM px.nextsample()");
}

```

#### 7.5.4.37 void lspg\_nextsample\_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 404 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));
}

```

#### 7.5.4.38 void lspg\_nextsample.done ( )

Called when the next shot query has been processed.

Definition at line 463 of file lspg.c.

```

    {
        pthread_mutex_unlock( &(lspg_nextsample.mutex));
    }

```

#### 7.5.4.39 void lspg\_nextsample.init ( )

Initialize the nextsample variable, mutex, and condition.

Definition at line 437 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);
    }

```

#### 7.5.4.40 void lspg\_nextsample.wait ( )

Wait for the nextsample query to get processed.

Definition at line 455 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));
    }

```

#### 7.5.4.41 void lspg\_nextshot.call ( )

Queue up a nextshot query.

Definition at line 824 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()");
}

```

#### 7.5.4.42 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	<i>qqp</i>	Our nextshot query
in	<i>pgr</i>	result of the query

Definition at line 569 of file lspg.c.

```

    {
static int got_col_nums=0;
static int
    dsdir_c, dspid_c, dsowidth_c, dsoscaxis_c, dsexp_c, skey_c, sstart_c, sfn_c
    , dsphi_c,
    dsomega_c, dskappa_c, dsdist_c, dsnrng_c, dshpid_c, cx_c, cy_c, ax_c, ay_c,
    az_c,
    active_c, sindex_c, stype_c,
    dsowidth2_c, dsoscaxis2_c, dsexp2_c, sstart2_c, dsphi2_c, dsomega2_c,
    dskappa2_c, dsdist2_c, dsnrng2_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");
    dspid_c = PQfnumber( pgr, "dspid");
    dsowidth_c = PQfnumber( pgr, "dsowidth");
    dsoscaxis_c = PQfnumber( pgr, "dsoscaxis");
    dsexp_c = PQfnumber( pgr, "dsexp");
    skey_c = PQfnumber( pgr, "skey");
    sstart_c = PQfnumber( pgr, "sstart");
    sfn_c = PQfnumber( pgr, "sfn");
    dsphi_c = PQfnumber( pgr, "dsphi");
    dsomega_c = PQfnumber( pgr, "dsomega");
    dskappa_c = PQfnumber( pgr, "dskappa");
    dsdist_c = PQfnumber( pgr, "dsdist");
    dsnrng_c = PQfnumber( pgr, "dsnrng");
    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");
    dsnrng2_c = PQfnumber( pgr, "dsnrng2");
}

```

```

    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.dmdir_isnull = PQgetisnull( pgr, 0,
    dmdir_c);
if( lspg_nextshot.dmdir != NULL)
    free( lspg_nextshot.dmdir);
lspg_nextshot.dmdir = strdup( PQgetvalue( pgr, 0, dmdir_c))
;

lspg_nextshot.dspid_isnull = PQgetisnull( pgr, 0,
    dspid_c);
if( lspg_nextshot.dspid != NULL)
    free( lspg_nextshot.dspid);
lspg_nextshot.dspid = strdup( PQgetvalue( pgr, 0, dspid_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( lspg_nextshot.dsowidth_isnull == 0)
    lspg_nextshot.dsowidth = atof( PQgetvalue( pgr,0,
    dsowidth_c));

lspg_nextshot.dsexp_isnull = PQgetisnull( pgr, 0,
    dsexp_c);
if( lspg_nextshot.dsexp_isnull == 0)
    lspg_nextshot.dsexp = atof( PQgetvalue( pgr,0, dsexp_c
    ));

lspg_nextshot.sstart_isnull = PQgetisnull( pgr, 0,
    sstart_c);
if( lspg_nextshot.sstart_isnull == 0)
    lspg_nextshot.sstart = atof( PQgetvalue( pgr,0,
    sstart_c));

lspg_nextshot.dsphi_isnull = PQgetisnull( pgr, 0,

```

```

    dsphi_c);
if( lspg_nextshot.dsphi_isnull == 0)
    lspg_nextshot.dsphi = atof( PQgetvalue( pgr,0, dsphi_c
    ));

lspg_nextshot.dsomega_isnull = PQgetisnull( pgr, 0
    , dsomega_c);
if( lspg_nextshot.dsomega_isnull == 0)
    lspg_nextshot.dsomega = atof( PQgetvalue( pgr,0,
    dsomega_c));

lspg_nextshot.dskappa_isnull = PQgetisnull( pgr, 0
    , dskappa_c);
if( lspg_nextshot.dskappa_isnull == 0)
    lspg_nextshot.dskappa = atof( PQgetvalue( pgr,0,
    dskappa_c));

lspg_nextshot.dsdist_isnull = PQgetisnull( pgr, 0,
    dsdist_c);
if( lspg_nextshot.dsdist_isnull == 0)
    lspg_nextshot.dsdist = atof( PQgetvalue( pgr,0,
    dsdist_c));

lspg_nextshot.dsnrg_isnull = PQgetisnull( pgr, 0,
    dsnrg_c);
if( lspg_nextshot.dsnrg_isnull == 0)
    lspg_nextshot.dsnrg = atof( PQgetvalue( pgr,0, dsnrg_c
    ));

lspg_nextshot.cx_isnull = PQgetisnull( pgr, 0, cx_c);
if( lspg_nextshot.cx_isnull == 0)
    lspg_nextshot.cx = atof( PQgetvalue( pgr,0, cx_c));

lspg_nextshot.cy_isnull = PQgetisnull( pgr, 0, cy_c);
if( lspg_nextshot.cy_isnull == 0)
    lspg_nextshot.cy = atof( PQgetvalue( pgr,0, cy_c));

lspg_nextshot.ax_isnull = PQgetisnull( pgr, 0, ax_c);
if( lspg_nextshot.ax_isnull == 0)
    lspg_nextshot.ax = atof( PQgetvalue( pgr,0, ax_c));

lspg_nextshot.ay_isnull = PQgetisnull( pgr, 0, ay_c);
if( lspg_nextshot.ay_isnull == 0)
    lspg_nextshot.ay = atof( PQgetvalue( pgr,0, ay_c));

lspg_nextshot.az_isnull = PQgetisnull( pgr, 0, az_c);
if( lspg_nextshot.az_isnull == 0)
    lspg_nextshot.az = atof( PQgetvalue( pgr,0, az_c));

lspg_nextshot.active_isnull = PQgetisnull( pgr, 0,
    active_c);
if( lspg_nextshot.active_isnull == 0)
    lspg_nextshot.active = atoi( PQgetvalue( pgr, 0,
    active_c));

lspg_nextshot.sindex_isnull = PQgetisnull( pgr, 0,
    sindex_c);
if( lspg_nextshot.sindex_isnull == 0)
    lspg_nextshot.sindex = atoi( PQgetvalue( pgr, 0,
    sindex_c));

lspg_nextshot.dshpid_isnull = PQgetisnull( pgr, 0,
    dshpid_c);
if( lspg_nextshot.dshpid_isnull == 0)
    lspg_nextshot.dshpid = atoi( PQgetvalue( pgr, 0,
    dshpid_c));

lspg_nextshot.skey_isnull = PQgetisnull( pgr, 0,
    skey_c);
if( lspg_nextshot.skey_isnull == 0)
    lspg_nextshot.skey = atoll( PQgetvalue( pgr, 0, skey_c)
    );

lspg_nextshot.dsowidth2_isnull = PQgetisnull(
    pgr, 0, dsowidth2_c);
if( lspg_nextshot.dsowidth2_isnull == 0)
    lspg_nextshot.dsowidth2 = atof( PQgetvalue( pgr,0,
    dsowidth2_c));

lspg_nextshot.dsexp2_isnull = PQgetisnull( pgr, 0,
    dsexp2_c);
if( lspg_nextshot.dsexp2_isnull == 0)
    lspg_nextshot.dsexp2 = atof( PQgetvalue( pgr,0,
    dsexp2_c));

lspg_nextshot.sstart2_isnull = PQgetisnull( pgr, 0
    , sstart2_c);

```

```

if( lspg_nextshot.sstart2_isnull == 0)
    lspg_nextshot.sstart2 = atof( PQgetvalue( pgr,0,
        sstart2_c));

lspg_nextshot.dsphi2_isnull = PQgetisnull( pgr, 0,
    dsphi2_c);
if( lspg_nextshot.dsphi2_isnull == 0)
    lspg_nextshot.dsphi2 = atof( PQgetvalue( pgr,0,
        dsphi2_c));

lspg_nextshot.dsomega2_isnull = PQgetisnull( pgr,
    0, dsomega2_c);
if( lspg_nextshot.dsomega2_isnull == 0)
    lspg_nextshot.dsomega2 = atof( PQgetvalue( pgr,0,
        dsomega2_c));

lspg_nextshot.dskappa2_isnull = PQgetisnull( pgr,
    0, dskappa2_c);
if( lspg_nextshot.dskappa2_isnull == 0)
    lspg_nextshot.dskappa2 = atof( PQgetvalue( pgr,0,
        dskappa2_c));

lspg_nextshot.dsdist2_isnull = PQgetisnull( pgr, 0
    , dsdist2_c);
if( lspg_nextshot.dsdist2_isnull == 0)
    lspg_nextshot.dsdist2 = atof( PQgetvalue( pgr,0,
        dsdist2_c));

lspg_nextshot.dsnrg2_isnull = PQgetisnull( pgr, 0,
    dsnrg2_c);
if( lspg_nextshot.dsnrg2_isnull == 0)
    lspg_nextshot.dsnrg2 = atof( PQgetvalue( pgr,0,
        dsnrg2_c));

lspg_nextshot.cx2_isnull = PQgetisnull( pgr, 0, cx2_c)
;
if( lspg_nextshot.cx2_isnull == 0)
    lspg_nextshot.cx2 = atof( PQgetvalue( pgr,0, cx2_c));

lspg_nextshot.cy2_isnull = PQgetisnull( pgr, 0, cy2_c)
;
if( lspg_nextshot.cy2_isnull == 0)
    lspg_nextshot.cy2 = atof( PQgetvalue( pgr,0, cy2_c));

lspg_nextshot.ax2_isnull = PQgetisnull( pgr, 0, ax2_c)
;
if( lspg_nextshot.ax2_isnull == 0)
    lspg_nextshot.ax2 = atof( PQgetvalue( pgr,0, ax2_c));

lspg_nextshot.ay2_isnull = PQgetisnull( pgr, 0, ay2_c)
;
if( lspg_nextshot.ay2_isnull == 0)
    lspg_nextshot.ay2 = atof( PQgetvalue( pgr,0, ay2_c));

lspg_nextshot.az2_isnull = PQgetisnull( pgr, 0, az2_c)
;
if( lspg_nextshot.az2_isnull == 0)
    lspg_nextshot.az2 = atof( PQgetvalue( pgr,0, az2_c));

lspg_nextshot.active2_isnull = PQgetisnull( pgr, 0
    , active2_c);
if( lspg_nextshot.active2_isnull == 0)
    lspg_nextshot.active2 = atoi( PQgetvalue( pgr, 0,
        active2_c));

lspg_nextshot.sindex2_isnull = PQgetisnull( pgr, 0
    , sindex2_c);
if( lspg_nextshot.sindex2_isnull == 0)
    lspg_nextshot.sindex2 = atoi( PQgetvalue( pgr, 0,
        sindex2_c));

lspg_nextshot.new_value_ready = 1;

pthread_cond_signal( &(lspg_nextshot.cond));
pthread_mutex_unlock( &(lspg_nextshot.mutex));
}

```

#### 7.5.4.43 void lspg\_nextshot.done ( )

Called when the next shot query has been processed.

Definition at line 842 of file lspg.c.

```

    {
        pthread_mutex_unlock( &(lspg_nextshot.mutex));
    }

```

#### 7.5.4.44 void lspg\_nextshot\_init ( )

Initialize the nextshot variable, mutex, and condition.

Definition at line 816 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);
    }

```

#### 7.5.4.45 void lspg\_nextshot\_wait ( )

Wait for the next shot query to get processed.

Definition at line 834 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));
    }

```

#### 7.5.4.46 PQnoticeProcessor lspg\_notice\_processor ( void \* arg, const char \* msg )

Definition at line 1527 of file lspg.c.

```

    {
        lslogging_log_message( "lspg: %s", msg);
        return NULL;
    }

```

#### 7.5.4.47 void lspg\_pg\_connect ( )

Connect to the pg server.

Definition at line 1534 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 the 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;
}
}

```

#### 7.5.4.48 void lspg\_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 1428 of file lspg.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) {
        lspg_connectPoll_response = PQconnectPoll( q);
        if( lspg_connectPoll_response ==
            PGRES_POLLING_FAILED) {
            ls_pg_state = LS_PG_STATE_RESET;
        }
        return;
    }

    if( ls_pg_state == LS_PG_STATE_RESET_POLL)
    {
        lspg_resetPoll_response = PQresetPoll( q);
        if( lspg_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) {
    lspg_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( "lspg_pg_service: notify
            recieved %s", pgn->relnam);

        if( strstr( pgn->relnam, "_pmac") != NULL) {
            lspg_query_push( lspg_cmd_cb, "SELECT
                pmac.md2_queue_next()");
        } else if (strstr( pgn->relnam, "_diff") != NULL || strstr( pgn->
            relnam, "_run") != NULL) {
            lspg_query_push( lspg_nextaction_cb,
                "SELECT action FROM px.nextaction()");
        } else if (strstr( pgn->relnam, "_sample") != NULL) {
            lspg_getcurrentsampleid_call();
        }
        PQfreemem( pgn);
    }
}

if( evt->revents & POLLOUT) {

    if( ls_pg_state == LS_PG_STATE_INIT_POLL) {
        lspg_connectPoll_response = PQconnectPoll( q);
        if( lspg_connectPoll_response ==
            PGRES_POLLING_FAILED) {
            ls_pg_state = LS_PG_STATE_RESET;
        }
        return;
    }
}

```

```

if( ls_pg_state == LS_PG_STATE_RESET_POLL)
{
    lspg_resetPoll_response = PQresetPoll( q );
    if( lspg_resetPoll_response ==
PGRES_POLLING_FAILED) {
        ls_pg_state = LS_PG_STATE_RESET;
    }
    return;
}

if( ls_pg_state == LS_PG_STATE_SEND) {
    lspg_send_next_query();
}

if( ls_pg_state == LS_PG_STATE_SEND_FLUSH)
{
    lspg_flush();
}
}
}

```

#### 7.5.4.49 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 75 of file lspg.c.

```

{
    lspg_query_queue_t *rtn;

    pthread_mutex_lock( &lspg_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_queue_cond);
    }
    pthread_mutex_unlock( &lspg_queue_mutex);

    return rtn;
}

```

#### 7.5.4.50 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 128 of file lspg.c.

```

{
    int idx;
    va_list arg_ptr;

    pthread_mutex_lock( &lspg_queue_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_queue_cond, &lspg_queue_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);
};

```

#### 7.5.4.51 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 99 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);
}

```

#### 7.5.4.52 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 112 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;
}

```

#### 7.5.4.53 void lspg\_receive ( )

Receive a result of a query.

Definition at line 1345 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 ) {
                lspg_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 = lspg_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 ) {
                        lslogging_log_message( "Error from query '%s':\n
%s", qqp->qid, 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 );
            }
        }
    }
}

```

#### 7.5.4.54 void lspg\_run ( )

Start 'er runnin'.

Definition at line 1777 of file lspg.c.

```

    {
        pthread_create( &lspg_thread, NULL, lspg_worker, NULL );
        lsevents_add_listener( "Sample(Detected|Absent)",
            lspmac_sample_detector_cb );
    }
}

```

#### 7.5.4.55 void lspg\_send\_next\_query ( )

send the next queued query to the DB server

Definition at line 1298 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;
}
}
}
}

```

**7.5.4.56** void `lspg_seq_run_prep_all` ( long long *skey*, double *kappa*, double *phi*, double *cx*, double *cy*, double *ax*, double *ay*, double *az* )

Convenience function to call seq run prep.

#### Parameters

in	<i>skey</i>	px.shots key for this image
in	<i>kappa</i>	current kappa postion
in	<i>phi</i>	current phi postition
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 1095 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();
}

```

**7.5.4.57** 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

#### Parameters

in	<i>skey</i>	px.shots key for this image
in	<i>kappa</i>	current kappa postion
in	<i>phi</i>	current phi postition
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 1061 of file `lspg.c`.

```

    {
        pthread_mutex_lock( &(lspg_seq_run_prep.mutex));
        lspg_seq_run_prep.new_value_ready = 0;
        pthread_mutex_unlock( &(lspg_seq_run_prep.mutex));

        lspg_query_push( lspg_seq_run_prep.cb, "
            SELECT px.seq_run_prep( %lld, %.3f, %.3f, %.3f, %.3f, %.3f, %.3f, %.3f)",
                skey, kappa, phi, cx, cy, ax, ay, az);
    }

```

**7.5.4.58** void `lspg_seq_run_prep_cb` ( `lspg_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 1049 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));
    }

```

**7.5.4.59** void `lspg_seq_run_prep_done` ( )

Indicate we are done waiting.

Definition at line 1089 of file `lspg.c`.

```

    {
        pthread_mutex_unlock( &(lspg_seq_run_prep.mutex));
    }

```

**7.5.4.60** void `lspg_seq_run_prep_init` ( )

Initialize the data collection object.

Definition at line 1041 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);
    }

```

#### 7.5.4.61 void lspg\_seq\_run\_prep.wait ( )

Wait for seq run prep query to return.

Definition at line 1081 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));
    }

```

#### 7.5.4.62 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 1406 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 involve 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));
    }

```

#### 7.5.4.63 int lspg\_starttransfer.all ( int \* err, unsigned int nextsample, int sampledected, double ax, double ay, double az, double horz, double vert, double esttime )

Definition at line 322 of file lspg.c.

```

    {
        int rtn;

        lspg_starttransfer_call( nextsample, sampledected,
            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;
}

```

#### 7.5.4.64 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 302 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, %d, %.3f, %.3f, %.3f, %.3f, %.3f, %.3f",
            nextsample, sample_detected, ax, ay, az, horz
        , vert, esttime);
}

```

#### 7.5.4.65 void lspg\_starttransfer\_cb ( lspg\_query\_queue\_t \* *qqp*, PGresult \* *pgr* )

##### Parameters

in	<i>qqp</i>	Our nextsample query
in	<i>pgr</i>	result of the query

Definition at line 281 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 = 0;
        lspg_starttransfer.starttransfer = 0;
    } else {
        lspg_starttransfer.no_rows_returned = 1;
        if( PQgetisnull( pgr, 0, 0) || strtol( PQgetvalue( pgr, 0, 0), NULL, 0) !=
            1)
            lspg_starttransfer.starttransfer = 0;
        else
            lspg_starttransfer.starttransfer = 1;
    }
    pthread_cond_signal( &(lspg_starttransfer.cond));
    pthread_mutex_unlock( &(lspg_starttransfer.mutex));
}

```

#### 7.5.4.66 void lspg\_starttransfer\_done ( )

Definition at line 317 of file lspg.c.

```

{
    pthread_mutex_unlock( &(lspg_starttransfer.mutex));
}

```

#### 7.5.4.67 void lspg\_starttransfer\_init ( )

Definition at line 275 of file lspg.c.

```

{
    lspg_starttransfer.new_value_ready = 0;
    pthread_mutex_init( &lspg_starttransfer.mutex, NULL);
    pthread_cond_init( &lspg_starttransfer.cond, NULL);
}

```

**7.5.4.68 void lspg\_starttransfer\_wait ( )**

Definition at line 311 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));
}

```

**7.5.4.69 void lspg\_wait\_for\_detector\_all ( )**

Combined call to wait for the detector.

Definition at line 905 of file lspg.c.

```

{
lspg_wait_for_detector_call();
lspg_wait_for_detector_wait();
lspg_wait_for_detector_done();
}

```

**7.5.4.70 void lspg\_wait\_for\_detector\_call ( )**

initiate the wait for detector query

Definition at line 879 of file lspg.c.

```

    {
pthread_mutex_lock( &(lspg_wait_for_detector.mutex
));
lspg_wait_for_detector.new_value_ready =
0;
pthread_mutex_unlock( &(lspg_wait_for_detector.mutex
));

lspg_query_push( lspg_wait_for_detector_cb
, "SELECT px.lock_detector_test_block()");
}

```

**7.5.4.71 void lspg\_wait\_for\_detector\_cb ( lspg\_query\_queue\_t \* qqp, PGresult \* pgr )**

Callback for the wait for detector query.

Definition at line 870 of file lspg.c.

```

    {
pthread_mutex_lock( &(lspg_wait_for_detector.mutex
));
lspg_wait_for_detector.new_value_ready =
1;
pthread_cond_signal( &(lspg_wait_for_detector.cond
));
pthread_mutex_unlock( &(lspg_wait_for_detector.mutex
));
}

```

**7.5.4.72 void lspg\_wait\_for\_detector\_done ( )**

Done waiting for the detector.

Definition at line 898 of file lspg.c.

```

        {
pthread_mutex_unlock( &(lspg_wait_for_detector.mutex
));
}

```

#### 7.5.4.73 void lspg\_wait\_for\_detector\_init ( )

initialize the detector timing object

Definition at line 862 of file lspg.c.

```

        {
lspg_wait_for_detector.new_value_ready =
0;
pthread_mutex_init( &(lspg_wait_for_detector.mutex
), NULL);
pthread_cond_init( &(lspg_wait_for_detector.cond),
NULL);
}

```

#### 7.5.4.74 void lspg\_wait\_for\_detector\_wait ( )

Pause the calling thread until the detector is ready Called by the MD2 thread.

Definition at line 890 of file lspg.c.

```

        {
pthread_mutex_lock( &(lspg_wait_for_detector.mutex
));
while( lspg_wait_for_detector.new_value_ready
== 0)
pthread_cond_wait( &(lspg_wait_for_detector.cond)
, &(lspg_wait_for_detector.mutex));
}

```

#### 7.5.4.75 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

Definition at line 507 of file lspg.c.

```

        {
pthread_mutex_lock( &lspg_waitcryo.mutex);
lspg_waitcryo.new_value_ready = 0;

lspg_query_push( lspg_waitcryo_cb, "SELECT
px.waitcryo()");

while( lspg_waitcryo.new_value_ready == 0)
pthread_cond_wait( &lspg_waitcryo.cond, &lspg_waitcryo
.mutex);

pthread_mutex_unlock( &lspg_waitcryo.mutex);
}

```

#### 7.5.4.76 void lspg\_waitcryo.cb ( lspg\_query\_queue\_t \* qqp, PGresult \* pgr )

Definition at line 497 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);
}

```



**7.5.4.77 void lspg\_waitcryo\_init ( )**

Definition at line 491 of file lspg.c.

```

    {
        lspg_waitcryo.new_value_ready = 0;
        pthread_mutex_init( &lspg_waitcryo.mutex, NULL);
        pthread_cond_init( &lspg_waitcryo.cond, NULL);
    }

```

**7.5.4.78 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 1674 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 conneceted
        //
        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.
                // Periodicaly try again. Should possibly arrange to reconnect
                // to signalfd but that's unlikely to be nessesary.
                //
                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--;
    }
}
}

```

#### 7.5.4.79 void lspmac\_sample\_detector\_cb ( char \* event )

log magnet state

Definition at line 1746 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");
}

```

### 7.5.5 Variable Documentation

#### 7.5.5.1 int ls\_pg\_state = LS\_PG\_STATE\_INIT [static]

State of the lspg state machine.

Definition at line 39 of file lspg.c.

#### 7.5.5.2 PostgresPollingStatusType lspg\_connectPoll\_response [static]

Used to determine state while connecting.

Definition at line 60 of file lspg.c.

#### 7.5.5.3 lspg\_demandairrights\_t lspg\_demandairrights

our demandairrights object

Definition at line 66 of file lspg.c.

#### 7.5.5.4 lspg\_getcenter\_t lspg\_getcenter

the getcenter object

Definition at line 65 of file lspg.c.

#### 7.5.5.5 lspg\_getcurrentsampleid\_t lspg\_getcurrentsampleid

our currentsample id

Definition at line 67 of file lspg.c.

#### 7.5.5.6 `lspg_lock_detector_t` `lspg_lock_detector` [static]

Definition at line 979 of file `lspg.c`.

#### 7.5.5.7 `lspg_lock_diffractionmeter_t` `lspg_lock_diffractionmeter` [static]

Definition at line 920 of file `lspg.c`.

#### 7.5.5.8 `lspg_nextsample_t` `lspg_nextsample`

the very next sample

Definition at line 63 of file `lspg.c`.

#### 7.5.5.9 `lspg_nextshot_t` `lspg_nextshot`

the nextshot object

Definition at line 64 of file `lspg.c`.

#### 7.5.5.10 `lspg_query_queue_t` `lspg_query_queue[LS_PG_QUERY_QUEUE_LENGTH]` [static]

Our query queue.

Definition at line 52 of file `lspg.c`.

#### 7.5.5.11 `unsigned int` `lspg_query_queue_off = 0` [static]

The last item still being used (on == off means nothing in queue)

Definition at line 54 of file `lspg.c`.

#### 7.5.5.12 `unsigned int` `lspg_query_queue_on = 0` [static]

Next position to add something to the queue.

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

#### 7.5.5.13 `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 55 of file `lspg.c`.

#### 7.5.5.14 `pthread_cond_t` `lspg_queue_cond` [static]

keeps the queue from overflowing

Definition at line 44 of file `lspg.c`.

**7.5.5.15** `pthread_mutex_t lspg_queue_mutex` `[static]`

keep the queue from getting tangled

Definition at line 43 of file lspg.c.

**7.5.5.16** `PostgresPollingStatusType lspg_resetPoll_response` `[static]`

Used to determine state while reconnecting.

Definition at line 61 of file lspg.c.

**7.5.5.17** `lspg_seq_run_prep_t lspg_seq_run_prep` `[static]`

Definition at line 1037 of file lspg.c.

**7.5.5.18** `lspg_starttransfer_t lspg_starttransfer`

start a sample transfer

Definition at line 68 of file lspg.c.

**7.5.5.19** `pthread_t lspg_thread` `[static]`

our worker thread

Definition at line 42 of file lspg.c.

**7.5.5.20** `lspg_wait_for_detector_t lspg_wait_for_detector` `[static]`

Instance of the detector timing object.

Definition at line 858 of file lspg.c.

**7.5.5.21** `lspg_waitcryo_t lspg_waitcryo`

signal the robot

Definition at line 69 of file lspg.c.

**7.5.5.22** `struct pollfd lspgfd` `[static]`

our poll info

Definition at line 45 of file lspg.c.

**7.5.5.23** `struct timeval lspg_time_sent now` `[static]`

used to ensure we do not inundate the db server with connection requests

Definition at line 40 of file lspg.c.

## 7.5.5.24 PGconn\* q = NULL [static]

Database connector.

Definition at line 59 of file lspg.c.

## 7.6 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](#)

### 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\\_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\\_SENDLINE](#) 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 100.0`  
*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`

## Functions

- `void lspmac_get_ascii (char *)`  
*Forward declarateion.*
- `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.*
- `void cleanstr (char *s)`  
*Replace \r with \n in null terminated string and print result to terminal.*
- `void lsConnect (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 wIndex, 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 upate 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\\_SockSendDPqueue](#) ()
- void [lspmac\\_GetAllIVarsCB](#) (lspmac\_cmd\_queue\_t \*cmd, int nreceived, char \*buff)  
*Receive the values of all the I variables Update our Postgresql database with the results.*
- void [lspmac\\_GetAllIVars](#) ()  
*Request the values of all the I variables.*
- void [lspmac\\_GetAllMVarsCB](#) (lspmac\_cmd\_queue\_t \*cmd, int nreceived, char \*buff)  
*Receive the values of all the M variables Update our database with the results.*
- void [lspmac\\_GetAllMVars](#) ()  
*Request the values of all the M variables.*
- void [lspmac\\_sendcmd\\_nocb](#) (char \*fmt,...)  
*Send a command that does not need to deal with the reply.*
- void [lspmac\\_sendcmd](#) (char \*event, void(\*responseCB)(lspmac\_cmd\_queue\_t \*, int, char \*), char \*fmt,...)  
*PMAC command with call back.*
- void [lspmac\\_next\\_state](#) ()  
*State machine logic.*
- void \* [lspmac\\_worker](#) (void \*dummy)  
*Our lspmac worker thread.*
- void [lspmac\\_movedac\\_queue](#) (lspmac\_motor\_t \*mp, double requested\_position)  
*Move method for dac motor objects (ie, lights)*
- void [lspmac\\_movezoom\\_queue](#) (lspmac\_motor\_t \*mp, double requested\_position)  
*Move method for the zoom motor.*
- void [lspmac\\_move\\_preset\\_queue](#) (lspmac\_motor\_t \*mp, char \*preset\_name)  
*Move a given motor to one of its preset positions.*
- int [lspmac\\_test\\_preset](#) (lspmac\_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*
- void [lspmac\\_moveabs\\_fshut\\_queue](#) (lspmac\_motor\_t \*mp, double requested\_position)  
*Move method for the fast shutter.*
- void [lspmac\\_moveabs\\_bo\\_queue](#) (lspmac\_motor\_t \*mp, double requested\_position)  
*Move method for binary i/o motor objects.*
- void [lspmac\\_moveabs\\_timed\\_queue](#) (lspmac\_motor\_t \*mp, double start, double delta, double time)  
*timed motor move*
- void [lspmac\\_moveabs\\_frontlight\\_oo\\_queue](#) (lspmac\_motor\_t \*mp, double pos)  
*"move" frontlight on/off*
- void [lspmac\\_moveabs\\_flight\\_factor\\_queue](#) (lspmac\_motor\_t \*mp, double pos)
- void [lspmac\\_moveabs\\_blight\\_factor\\_queue](#) (lspmac\_motor\_t \*mp, double pos)
- void [lspmac\\_video\\_rotate](#) (double secs)  
*Special motion program to collect centering video.*
- void [lspmac\\_move\\_or\\_jog\\_abs\\_queue](#) (lspmac\_motor\_t \*mp, double requested\_position, int use\_jog)  
*Move method for normal stepper and servo motor objects.*
- void [lspmac\\_move\\_or\\_jog\\_preset\\_queue](#) (lspmac\_motor\_t \*mp, char \*preset, int use\_jog)  
*move using a preset value*
- void [lspmac\\_moveabs\\_queue](#) (lspmac\_motor\_t \*mp, double requested\_position)  
*Use coordinate system motion program, if available, to move motor to requested position.*
- void [lspmac\\_jogabs\\_queue](#) (lspmac\_motor\_t \*mp, double requested\_position)  
*Use jog to move motor to requested position.*
- void [lspmac\\_moveabs\\_wait](#) (lspmac\_motor\_t \*mp)  
*Wait for motor to finish moving.*
- void [\\_lspmac\\_motor\\_init](#) (lspmac\_motor\_t \*d, char \*name)  
*Helper functon for the init calls.*
- [lspmac\\_motor\\_t](#) \* [lspmac\\_motor\\_init](#) (lspmac\_motor\_t \*d, int wy, int wx, int \*posp, int \*stat1p, int \*stat2p, char \*wtitle, char \*name, void(\*moveAbs)(lspmac\_motor\_t \*, double))



- Initialize a pmac stepper or servo motor.*
  - `lspmac_motor_t * lspmac_fshut_init (lspmac_motor_t *d)`
- Initialize the fast shutter motor.*
  - `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.*
  - `lspmac_motor_t * lspmac_dac_init (lspmac_motor_t *d, int *posp, char *mvar, char *name, void(*moveAbs)(lspmac_motor_t *, double))`
- Initialize DAC motor Note that some motors require further initialization from a database query.*
  - `void lspmac_soft_motor_read (lspmac_motor_t *p)`
- Dummy routine to read a soft motor.*
  - `lspmac_motor_t * lspmac_soft_motor_init (lspmac_motor_t *d, char *name, void(*moveAbs)(lspmac_motor_t *, double))`
- `lspmac_bi_t * lspmac_bi_init (lspmac_bi_t *d, int *ptr, int mask, char *onEvent, char *offEvent)`
- Initialize binary input.*
  - `void lspmac_init (int ivarsflag, int mvarsflag)`
- Initialize this module.*
  - `void lspmac_cryoSwitchChanged_cb (char *event)`
  - `void lspmac_scint_inPosition_cb (char *event)`
- Maybe start drying off the scintillator.*
  - `void lspmac_backLight_up_cb (char *event)`
- Turn on the backlight whenever it goes up.*
  - `void lspmac_backLight_down_cb (char *event)`
- Turn off the backlight whenever it goes down.*
  - `void lspmac_light_zoom_cb (char *event)`
- Set the backlight intensity whenever the zoom is changed (and the backlight is up)*
  - `void lspmac_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.*
  - `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 lsredis_obj_t * lspmac_md2_init`
- `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 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](#)  
*our thread to manage access and communication to the pmac*
- 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*
- 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](#) [48]  
*All our motors.*
- int [lspmac\\_nmotors](#) = 0  
*The number of motors we manage.*
- [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 X.*
- [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 scintillator 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

### 7.6.1 Detailed Description

Routines concerned with communication with PMAC.

```
\date 2012
\author Keith Brister
\copyright All Rights Reserved
```

This is a state machine (surprise!) Lacking is 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.

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

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

### 7.6.2 Macro Definition Documentation

#### 7.6.2.1 #define LS\_PMAC\_STATE\_CR 7

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

#### 7.6.2.2 #define LS\_PMAC\_STATE\_DETACHED 0

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

#### 7.6.2.3 #define LS\_PMAC\_STATE\_GB 10

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

#### 7.6.2.4 `#define LS_PMAC_STATE_GMR 6`

Definition at line 44 of file `lspmac.c`.

#### 7.6.2.5 `#define LS_PMAC_STATE_IDLE 1`

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

#### 7.6.2.6 `#define LS_PMAC_STATE_RESET -1`

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

#### 7.6.2.7 `#define LS_PMAC_STATE_RR 8`

Definition at line 46 of file `lspmac.c`.

#### 7.6.2.8 `#define LS_PMAC_STATE_SC 2`

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

#### 7.6.2.9 `#define LS_PMAC_STATE_WACK 5`

Definition at line 43 of file `lspmac.c`.

#### 7.6.2.10 `#define LS_PMAC_STATE_WACK_CC 4`

Definition at line 42 of file `lspmac.c`.

#### 7.6.2.11 `#define LS_PMAC_STATE_WACK_NFR 3`

Definition at line 41 of file `lspmac.c`.

#### 7.6.2.12 `#define LS_PMAC_STATE_WACK_RR 9`

Definition at line 47 of file `lspmac.c`.

#### 7.6.2.13 `#define LS_PMAC_STATE_WCR 11`

Definition at line 49 of file `lspmac.c`.

#### 7.6.2.14 `#define LS_PMAC_STATE_WGB 12`

Definition at line 50 of file `lspmac.c`.

#### 7.6.2.15 `#define LSPMAC_DPASCII_QUEUE_LENGTH 1024`

Definition at line 355 of file `lspmac.c`.

**7.6.2.16** `#define LSPMAC_PRESET_REGEX "(.*\\.%s\\.presets)\\.([0-9]+)\\.\\. (name|position)"`

Regex to pick out preset name and corresponding position.

Definition at line 138 of file lspmac.c.

**7.6.2.17** `#define PMAC_CMD_QUEUE_LENGTH 2048`

Size of the PMAC command queue.

Definition at line 182 of file lspmac.c.

**7.6.2.18** `#define pmac_cmd_size 8`

PMAC command size in bytes.

Definition at line 148 of file lspmac.c.

**7.6.2.19** `#define PMAC_MIN_CMD_TIME 100.0`

Minimum time between commands to the pmac.

Definition at line 178 of file lspmac.c.

**7.6.2.20** `#define PMACPORT 1025`

The PMAC (only) listens on this port.

Definition at line 142 of file lspmac.c.

**7.6.2.21** `#define VR_CTRL_RESPONSE 0xc4`

Definition at line 164 of file lspmac.c.

**7.6.2.22** `#define VR_DOWNLOAD 0x40`

Definition at line 151 of file lspmac.c.

**7.6.2.23** `#define VR_FWDOWNLOAD 0xcb`

Definition at line 168 of file lspmac.c.

**7.6.2.24** `#define VR_IPADDRESS 0xe0`

Definition at line 169 of file lspmac.c.

**7.6.2.25** `#define VR_PMAC_FLUSH 0xb3`

Definition at line 155 of file lspmac.c.

**7.6.2.26** `#define VR_PMAC_GETBUFFER 0xc5`

Definition at line 165 of file lspmac.c.

**7.6.2.27 #define VR\_PMAC\_GETLINE 0xb1**

Definition at line 154 of file lspmac.c.

**7.6.2.28 #define VR\_PMAC\_GETMEM 0xb4**

Definition at line 156 of file lspmac.c.

**7.6.2.29 #define VR\_PMAC\_GETRESPONSE 0xbf**

Definition at line 162 of file lspmac.c.

**7.6.2.30 #define VR\_PMAC\_PORT 0xbe**

Definition at line 161 of file lspmac.c.

**7.6.2.31 #define VR\_PMAC\_READREADY 0xc2**

Definition at line 163 of file lspmac.c.

**7.6.2.32 #define VR\_PMAC\_SENDCTRLCHAR 0xb6**

Definition at line 158 of file lspmac.c.

**7.6.2.33 #define VR\_PMAC\_SENDLINE 0xb0**

Definition at line 153 of file lspmac.c.

**7.6.2.34 #define VR\_PMAC\_SETBIT 0xba**

Definition at line 159 of file lspmac.c.

**7.6.2.35 #define VR\_PMAC\_SETBITS 0xbb**

Definition at line 160 of file lspmac.c.

**7.6.2.36 #define VR\_PMAC\_SETMEM 0xb5**

Definition at line 157 of file lspmac.c.

**7.6.2.37 #define VR\_PMAC\_WRITEBUFFER 0xc6**

Definition at line 166 of file lspmac.c.

**7.6.2.38 #define VR\_PMAC\_WRITEERROR 0xc7**

Definition at line 167 of file lspmac.c.



## 7.6.2.39 #define VR\_UPLOAD 0xc0

Definition at line 150 of file Ispmac.c.

## 7.6.3 Typedef Documentation

## 7.6.3.1 typedef struct Ispmac\_ascii\_buffers\_struct Ispmac\_ascii\_buffers\_t

## 7.6.3.2 typedef struct Ispmac\_dpascii\_queue\_struct Ispmac\_dpascii\_queue\_t

## 7.6.3.3 typedef struct md2StatusStruct md2\_status\_t

The block of memory retrieved in a status request.

## 7.6.4 Function Documentation

## 7.6.4.1 void Ispmac\_motor\_init( Ispmac\_motor\_t \* d, char \* name )

Helper function for the init calls.

Definition at line 2726 of file Ispmac.c.

```

{
    lspmac_nmotors++;

    pthread_mutex_init( &(d->mutex), NULL);
    pthread_cond_init( &(d->cond), NULL);

    d->name = strdup(name);
    d->redis_position = lsredis_get_obj( "
        %s.position", d->name);
    d->u2c = lsredis_get_obj( "%s.u2c",
        d->name);
    d->printf_fmt = lsredis_get_obj( "
        %s.printf", d->name);
    d->redis_fmt = lsredis_get_obj( "%s.format",
        d->name);
    d->unit = lsredis_get_obj( "%s.unit",
        d->name);
    d->max_speed = lsredis_get_obj( "
        %s.max_speed", d->name);
    d->max_accel = lsredis_get_obj( "
        %s.max_accel", d->name);
    d->motor_num = lsredis_get_obj( "
        %s.motor_num", d->name);
    d->coord_num = lsredis_get_obj( "
        %s.coord_num", d->name);
    d->axis = lsredis_get_obj( "%s.axis",
        d->name);
    d->home = lsredis_get_obj( "%s.home",
        d->name);
    d->active = lsredis_get_obj( "%s.active",
        d->name);
    d->active_init = lsredis_get_obj( "
        %s.active_init", d->name);
    d->inactive_init = lsredis_get_obj( "
        %s.inactive_init", d->name);
    d->update_resolution = lsredis_get_obj( "
        %s.update_resolution", d->name);
    d->status_str = lsredis_get_obj( "
        %s.status_str", d->name);
    d->lut = NULL;
    d->nlut = 0;
    d->homing = 0;
    d->dac_mvar = NULL;
    d->actual_pos_cnts_p = NULL;
    d->status1_p = NULL;
    d->status2_p = NULL;
    d->win = NULL;
    d->read = NULL;
    d->reported_position = INFINITY;
}

```

#### 7.6.4.2 void 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 535 of file lspmac.c.

```

    {
        int i;

        pthread_mutex_lock( &ncurses_mutex);

        for( i=0; i<strlen( s); i++) {
            if( s[i] == '\r')
                wprintw( term_output, "\n");
            else
                wprintw( term_output, "%c", s[i]);
        }

        pthread_mutex_unlock( &ncurses_mutex);
    }

```

#### 7.6.4.3 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 508 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);
        }
    }

```

#### 7.6.4.4 void lsConnect ( char \* ipaddr )

Connect to the PMAC socket.

Establish or reestablish communications.

## Parameters

in	<i>ipaddr</i>	String representation of the IP address (dot quad or FQN)
----	---------------	---

Definition at line 556 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 = psock;
        pmacfd.events = POLLIN;
    }

```

#### 7.6.4.5 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 1904 of file lspmac.c.

```

    {
        lspmac_get_ascii( cmd->event);
    }

```

#### 7.6.4.6 void lspmac\_backLight\_down\_cb ( char \* event )

Turn off the backlight whenever it goes down.

## Parameters

<i>event</i>	Name of the event that called us
--------------	----------------------------------

Definition at line 3073 of file lspmac.c.

```

{
    blight->moveAbs( blight, 0.0);
}

```

#### 7.6.4.7 void lspmac\_backLight\_up.cb ( char \* event )

Turn on the backlight whenever it goes up.

## Parameters

<i>event</i>	Name of the event that called us
--------------	----------------------------------

Definition at line 3066 of file lspmac.c.

```

{
    blight->moveAbs( blight, round(lspmac_getPosition
        ( zoom)));
}

```

#### 7.6.4.8 lspmac\_bi\_t\* lspmac\_bi.init ( lspmac\_bi\_t \* d, int \* ptr, int mask, char \* onEvent, char \* offEvent )

Initialize binary input.

Definition at line 2875 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;

    return d;
}

```

#### 7.6.4.9 void lspmac\_blight\_lut\_setup ( )

Set up lookup table for blight.

Definition at line 3175 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.blight 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);
}

```

#### 7.6.4.10 `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 2809 of file `lspmac.c`.

```

{
    _lspmac_motor_init( d, name);

    d->moveAbs = 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;

    return d;
}

```

#### 7.6.4.11 `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 1119 of file `lspmac.c`.

```

{

```

```

int pos, changed;

pthread_mutex_lock( &(mp->mutex));

pos = (*(mp->read_ptr) & mp->read_mask) == 0 ? 0 : 1;

changed = pos != mp->position;
mp->position = pos;

pthread_mutex_unlock( &(mp->mutex));

if( changed)
    lsevents_send_event( "%s %d", mp->name, pos);
}

```

#### 7.6.4.12 void lspmac\_command\_done\_cb ( char \* event )

Definition at line 3235 of file lspmac.c.

```

{
int i;
lspmac_motor_t *mp;

// O(n). Bad.
//
for( i=0; i<lspmac_nmotors; i++) {
    if( strcmp( lspmac_motors[i].name, event, strlen(
        lspmac_motors[i].name)) == 0)
        break;
}

if( i >= lspmac_nmotors)
    return;

mp = &(lspmac_motors[i]);
pthread_mutex_lock( &(mp->mutex));

mp->command_sent = 1;

pthread_cond_signal( &(mp->cond));
pthread_mutex_unlock( &(mp->mutex));

return;
}

```

#### 7.6.4.13 void lspmac\_cryoSwitchChanged\_cb ( char \* event )

Definition at line 3028 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);
}

```

#### 7.6.4.14 lspmac\_motor\_t\* lspmac\_dac\_init ( lspmac\_motor\_t \* d, int \* posp, char \* mvar, char \* name, void (\*)(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

out	<i>d</i>	Returns the (almost) initialized motor object [in,out] unitintialized motor
in	<i>posp</i>	Location of current position
in	<i>mvar</i>	M variable, ie, "M1200"
in	<i>name</i>	name to coordinate with DB
in	<i>moveAbs</i>	Method to use to move this motor

Definition at line 2835 of file lspmac.c.

```

{
    _lspmac_motor_init( d, name);
    d->moveAbs          = moveAbs;
    d->read              = lspmac_dac_read;
    d->actual_pos_cnts_p = posp;
    d->dac_mvar          = strdup( mvar );

    return d;
}

```

#### 7.6.4.15 void lspmac\_dac\_read ( lspmac\_motor\_t \* mp )

Read a DAC motor position.

## Parameters

in	<i>mp</i>	The motor
----	-----------	-----------

Definition at line 1139 of file lspmac.c.

```

{
    double u2c;

    pthread_mutex_lock( &(mp->mutex));
    mp->actual_pos_cnts = *mp->actual_pos_cnts_p;
    u2c = lsredis_getd( mp->u2c );

    if( mp->nlut > 0 && mp->lut != NULL ) {
        if( u2c == 0.0 )
            u2c = 1.0;
        mp->position = lspmac_rlut( mp->nlut, mp->lut, mp
            ->actual_pos_cnts/u2c );
    } else {
        if( u2c != 0.0 ) {
            mp->position = mp->actual_pos_cnts / u2c;
        } else {
            mp->position = mp->actual_pos_cnts;
        }
    }

    pthread_mutex_unlock( &(mp->mutex));
}

```

#### 7.6.4.16 void lspmac\_Error ( char \* buff )

The service routing detected an error condition.

Scan the response buffer for an error code and print it out.

## Parameters

in	<i>buff</i>	Buffer returned by PMAC perhaps containing a NULL terminated message.
----	-------------	---

Definition at line 765 of file lspmac.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]);

        pthread_mutex_lock( &ncurses_mutex);
        wprintw( term_output, "\n%s\n", pmac_error_strs
                 [err]);
        wnoutrefresh( term_output);
        wnoutrefresh( term_input);
        doupdate();
        pthread_mutex_unlock( &ncurses_mutex);
    }
}
lspmac_Reset();
}

```

#### 7.6.4.17 void lspmac\_flight\_lut\_setup ( )

Set up lookup table for flight.

Definition at line 3142 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.flight 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);
}

```

#### 7.6.4.18 void lspmac\_fscint\_lut\_setup ( )

Set up lookup table for fscint.

Definition at line 3216 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);
}

```

#### 7.6.4.19 lspmac\_motor\_t\* lspmac\_fshut\_init( lspmac\_motor\_t \* d )

Initialize the fast shutter motor.

##### Parameters

in	<i>d</i>	Our uninitialized motor object
----	----------	--------------------------------

Definition at line 2793 of file lspmac.c.

```

{
    _lspmac_motor_init( d, "fastShutter");

    d->moveAbs          = lspmac_moveabs_fshut_queue
    ;
    d->read              = lspmac_shutter_read;

    return d;
}

```

#### 7.6.4.20 void lspmac\_get\_ascii( char \* event )

Forward declaretion.

Request the ascii buffers from the PMAC.

Definition at line 1896 of file lspmac.c.

```

{
    lspmac_send_command( VR_UPLOAD, VR_PMAC_GETMEM
        , 0x0e9c, 0, sizeof(lspmac_ascii_buffers_t), NULL,
        lspmac_get_ascii_cb, 0, event);
}

```

#### 7.6.4.21 void lspmac\_get\_ascii.cb( pmac\_cmd\_queue\_t \* cmd, int nreceived, char \* buff )

service the ascii buffer request response

Definition at line 1788 of file lspmac.c.

```

{
    uint32_t clrdata;
    int need_more;

    need_more = 0;
    pthread_mutex_lock( &lspmac_ascii_mutex);
    memcpy( &lspmac_ascii_buffers, buff, sizeof(
        lspmac_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( lspmac_ascii_buffers.response_buf == 0) {
    need_more = 1;
} else {
    if( (lspmac_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 = lspmac_ascii_buffers.response_buf
            & 0x000f;
        bcd2 = (lspmac_ascii_buffers.response_buf
            & 0x00f0) >> 4;
        bcd3 = (lspmac_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( "lspmac_get_ascii_cb: Error
            returned for %s: %s", lspmac_ascii_buffers.command_str
            , pmac_error_strs[errcode]);
        //
        // Command not allowed during program execution.
        //
        // Requeue it;
        if( errcode == 1) {
            lspmac_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)."
```

```

}

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."
//
clrdata = 0;          // set the control word to zero

if( need_more) {
    lspmac_send_command( VR_UPLOAD, VR_PMAC_SETMEM
        , 0x0f40, 0, 4, (char *)&clrdata, lspmac_more_ascii_cb, 1,
        NULL);
} else {
    lspmac_send_command( VR_UPLOAD, VR_PMAC_SETMEM
        , 0x0f40, 0, 4, (char *)&clrdata, NULL, 1, NULL);
    lspmac_ascii_busy = 0;
}
}

```

#### 7.6.4.22 void lspmac\_get\_status ( )

Request a status update from the PMAC.

Definition at line 1776 of file lspmac.c.

```

{
    lspmac_send_command( VR_UPLOAD, VR_PMAC_GETMEM
        , 0x400, 0, sizeof(md2_status_t), NULL, lspmac_get_status_cb
        , 0, NULL);
}

```

#### 7.6.4.23 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 1577 of file lspmac.c.

```

{
    static int cnt = 0;
    static struct timeval ts1;

    int i;
    lspmac_bi_t *bp;

    clock_gettime( CLOCK_REALTIME, &lspmac_status_time);

    if( cnt == 0) {
        gettimeofday( &ts1, NULL);
    }

    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
) {
    lslogging_log_message( "lspmac_get_status_cb: new
        moving flag: %0x", md2_status.moving_flags);
    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
// 0x40  6    M1006   Cryo is back

//
// acc11c_2    INPUTS
// mask  bit
// 0x01  0    M1008   Fluor Dector 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 Detected");
    else
        mvwprintw( term_status2, 2, 1, "%s", -(LS_DISPLAY_WINDOW_WIDTH
        -2), "Cap Not Detected");
    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);

/*
if( ++cnt % 1000 == 0) {
    gettimeofday( &ts2, NULL);

    lslogging_log_message( "Refresh Rate: %0.1f Hz", 1000000.*(cnt)/(ts2.tv_sec
        *1000000 + ts2.tv_usec - ts1.tv_sec*1000000 - ts1.tv_usec));

    cnt = 0;
}
*/
}

```

#### 7.6.4.24 void lspmac.GetAllIVars ( )

Request the values of all the I variables.

Definition at line 1976 of file lspmac.c.

```

{
static char *cmds = "I0..8191";
lspmac_send_command( VR_DOWNLOAD,
    VR_PMAC_SENDLINE, 0, 0, strlen( cmds), cmds,
    lspmac_GetAllIVarsCB, 0, NULL);
}

```

#### 7.6.4.25 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 1959 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);
}
}

```

#### 7.6.4.26 void lspmac.GetAllMVars ( )

Request the values of all the M variables.

Definition at line 2001 of file lspmac.c.

```

{
static char *cmds = "M0..8191->";
lspmac_send_command( VR_DOWNLOAD,
    VR_PMAC_SENDLINE, 0, 0, strlen( cmds), cmds,
    lspmac_GetAllMVarsCB, 0, NULL);
}

```

**7.6.4.27 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.

**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 1984 of file Ispmac.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);
}
}

```

**7.6.4.28 int Ispmac\_getBIPosition ( Ispmac\_bi\_t \* bip )**

get binary input value

Definition at line 1565 of file Ispmac.c.

```

    {
int rtn;
pthread_mutex_lock( &bip->mutex);
rtn = bip->position;
pthread_mutex_unlock( &bip->mutex);
return rtn;
}

```

**7.6.4.29 void Ispmac\_Getmem ( )**

Request a block of double buffer memory.

Definition at line 1110 of file Ispmac.c.

```

    {
int nbytes;
nbytes = (dbmemIn + 1400 > sizeof( dbmem)) ? sizeof( dbmem)
    - dbmemIn : 1400;
Ispmac_SockGetmem( dbmemIn, nbytes);
}

```

**7.6.4.30 void Ispmac\_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 recieved

Definition at line 1036 of file Ispmac.c.

```

{

memcpy( &(dbmem[ntohs(cmd->pcmd.wValue)]), buff, nreceived);

dbmemIn += nreceived;
if( dbmemIn >= sizeof( dbmem)) {
    dbmemIn = 0;
}
}
}

```

#### 7.6.4.31 double lspmac\_getPosition ( lspmac\_motor\_t \* mp )

get the motor position (with locking)

##### Parameters

<i>mp</i>	the motor object
-----------	------------------

Definition at line 1336 of file lspmac.c.

```

{

double rtn;
pthread_mutex_lock( &(mp->mutex));
rtn = mp->position;
pthread_mutex_unlock( &(mp->mutex));
return rtn;
}

```

#### 7.6.4.32 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 979 of file lspmac.c.

```

{

char *sp;    // pointer to the command this is a reply to

if( nreceived < 1400)
    buff[nreceived]=0;

sp = (char *) (cmd->pcmd.bData);

if( *buff == 0) {
    pthread_mutex_lock( &ncurses_mutex);
    wprintw( term_output, "%s\n", sp);
    pthread_mutex_unlock( &ncurses_mutex);
} else {
    pthread_mutex_lock( &ncurses_mutex);
    wprintw( term_output, "%s: ", sp);
    pthread_mutex_unlock( &ncurses_mutex);
    cleanstr( buff);
}
wnoutrefresh( term_output);
wnoutrefresh( term_input);
doupdate();

memset( cmd->pcmd.bData, 0, sizeof( cmd->pcmd.bData));
}

```



## 7.6.4.33 void lspmac\_home1\_queue( lspmac\_motor\_t \* mp )

Home the motor.

## Parameters

in	<i>mp</i>	motor we are concerned about
----	-----------	------------------------------

Definition at line 1203 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->homing) {
        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].homing || ((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->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_SockSendDPLine( mp->name, "%d$",
        motor_num);
}

pthread_mutex_unlock( &(mp->mutex));
}

```

#### 7.6.4.34 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 1289 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++) {

    pthread_mutex_lock( &ncurses_mutex);
    wprintw( term_output, "home2 is queuing '%s'\n", *spp);
    wnoutrefresh( term_output);
    doupdate();
    pthread_mutex_unlock( &ncurses_mutex);

    lspmac_SockSendDPLine( mp->name, *spp);
}

mp->homing = 2;
pthread_mutex_unlock( &(mp->mutex));
}

```

#### 7.6.4.35 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 2891 of file lspmac.c.

```

md2_status_t *p;
{
    // Set our global harvest flags
    getivars = ivarsflag;
    getmvars = mvarsflag;

    // All important status mutex
    pthread_mutex_init( &md2_status_mutex, NULL);

    //
    // Get the MD2 initialization strings
    //
    lspmac_md2_init = lsredis_get_obj( "
        md2_pmac.init");

    //
    // 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 A", "omega",
    lspmac_moveabs_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);
    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);
    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);
    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);
    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);
    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);
    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);
    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);
    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);
    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);
    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);
    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);
    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);

```

```

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);

fshut    = lspmac_fshut_init( &(lspmac_motors
[15]));
flight   = lspmac_dac_init( &(lspmac_motors[1
6]), &p->front_dac,      "M1200", "frontLight.intensity",
      lspmac_movedac_queue);
blight   = lspmac_dac_init( &(lspmac_motors[1
7]), &p->back_dac,      "M1201", "backLight.intensity",
      lspmac_movedac_queue);
fscint   = lspmac_dac_init( &(lspmac_motors[1
8]), &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_1), 0x01, "Fluor. Det. Parked",
"Fluor. Det. Not Parked");
sample_detected = lspmac_bi_init( &(lspmac_bis
[ 7]), &(md2_status.accllc_2), 0x02, "SamplePresent",
"SampleAbsent");
etel_ready   = lspmac_bi_init( &(lspmac_bis
[ 8]), &(md2_status.accllc_2), 0x20, "ETEL Ready",
"ETEL Not Ready");
etel_on      = lspmac_bi_init( &(lspmac_bis
[ 9]), &(md2_status.accllc_2), 0x40, "ETEL On",
"ETEL Off");
etel_init_ok  = lspmac_bi_init( &(lspmac_bis
[10]), &(md2_status.accllc_2), 0x80, "ETEL Init OK",
"ETEL Init Not OK");
minikappa_ok = lspmac_bi_init( &(lspmac_bis
[11]), &(md2_status.accllc_3), 0x01, "Minikappa OK",
"Minikappa Not OK");
smart_mag_on  = lspmac_bi_init( &(lspmac_bis
[12]), &(md2_status.accllc_3), 0x04, "Smart Magnet On",
"Smart Magnet Not On");
arm_parked    = lspmac_bi_init( &(lspmac_bis
[13]), &(md2_status.accllc_3), 0x08, "Arm Parked",
"Arm Not Parked");
smart_mag_err = lspmac_bi_init( &(lspmac_bis
[14]), &(md2_status.accllc_3), 0x10, "Smart Magnet Error",
"Smart Magnet OK");

```

```

shutter_open    = lspmac_bi_init( &(lspmac_bis
[15]), &(md2_status.acc11c_3), 0x100, "Shutter Open",
"Shutter Not Open");
smart_mag_off   = lspmac_bi_init( &(lspmac_bis
[16]), &(md2_status.acc11c_5), 0x01, "Smart Magnet Off",
"Smart Magnet Not Off");

//
// 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 mutexs and conditions
//

pthread_mutex_init( &pmac_queue_mutex, NULL);
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, NULL);
pthread_cond_init( &lspmac_shutter_cond, NULL);
pmacfd.fd = -1;

pthread_mutex_init( &lspmac_moving_mutex, NULL);
pthread_cond_init( &lspmac_moving_cond, NULL);

pthread_mutex_init( &lspmac_ascii_mutex, NULL);

pthread_mutex_init( &lspmac_ascii_buffers_mutex,
    NULL);

//
// 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");
}

```

#### 7.6.4.36 void 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 2665 of file lspmac.c.

```

{
    lspmac_move_or_jog_abs_queue( mp,
        requested_position, 1);
}
```

#### 7.6.4.37 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 3080 of file lspmac.c.

```

{
    int z;

    z = round(lspmac_getPosition( zoom));

    lslogging_log_message( "lspmac_light_zoom_cb: zoom = %d"
        , z);

    if( lspmac_getPosition( flight_oo) != 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);
    }
}
```

#### 7.6.4.38 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 376 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;
}

```

#### 7.6.4.39 void lspmac\_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 1782 of file lspmac.c.

```

{
    lspmac_get_ascii( cmd->event);
}

```

#### 7.6.4.40 lspmac\_motor\_t\* lspmac\_motor\_init( lspmac\_motor\_t \* d, int wy, int wx, int \* posp, int \* stat1p, int \* stat2p, char \* wtitle, char \* name, void(\*) (lspmac\_motor\_t \*, double) moveAbs )

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

Definition at line 2764 of file lspmac.c.

```

{

    _lspmac_motor_init( d, name);

    d->moveAbs          = moveAbs;
    d->read              = lspmac_pmacmotor_read;
    d->actual_pos_cnts_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);
    box( d->win, 0, 0);
    mvwprintw( d->win, 1, 1, "%s", wtitle);
    wnoutrefresh( d->win);

    return d;
}

```

**7.6.4.41** void lspmac\_move\_or\_jog\_abs\_queue ( lspmac\_motor\_t \* mp, double requested\_position, int use\_jog )

Move method for normal stepper and servo motor objects.

< 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 2515 of file lspmac.c.

```

{

    char *fmt;
    int q100;
    int requested_pos_cnts;
    int coord_num, motor_num;
    char *axis;
    double u2c;

    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);

    if( u2c == 0.0) {
        //
        // Shouldn't try moving a motor that has no units defined
        //
        pthread_mutex_unlock( &(mp->mutex));
        return;
    }
    mp->requested_position = requested_position;
    mp->not_done           = 1;
    mp->motion_seen        = 0;
    mp->command_sent       = 0;
    mp->requested_pos_cnts = u2c * requested_position;
    requested_pos_cnts    = mp->requested_pos_cnts;

    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
    // TODO: put in a timeout so we have a way out if something goes wrong
    // TODO: are we sure this thread is not the one moving it?
    //
    pthread_mutex_lock( &lspmac_moving_mutex);
    lslogging_log_message( "lspmac_moveabs_queue: waiting
        for previous moves to end.  lspmac_moving_flags = %0x", lspmac_moving_flags
    );
    while( (lspmac_moving_flags & q100) != 0)
        pthread_cond_wait( &lspmac_moving_cond, &
            lspmac_moving_mutex);
    pthread_mutex_unlock( &lspmac_moving_mutex);
    lslogging_log_message( "lspmac_moveabs_queue: Done.
        lspmac_moving_flags = %0x", lspmac_moving_flags);

    //
    // 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
    //
    pthread_mutex_lock( &lspmac_moving_mutex);
    lslogging_log_message( "lspmac_moveabs_queue: waiting
        for moving flag to propagate.  lspmac_moving_flags = %0x", lspmac_moving_flags
    );
    while( (lspmac_moving_flags & q100) == 0)
        pthread_cond_wait( &lspmac_moving_cond, &
            lspmac_moving_mutex);
    pthread_mutex_unlock( &lspmac_moving_mutex);
    lslogging_log_message( "lspmac_moveabs_queue: Done.
        lspmac_moving_flags = %0x", lspmac_moving_flags);
}
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);
}

```

#### 7.6.4.42 void lspmac\_move\_or\_jog\_preset\_queue ( lspmac\_motor\_t \* mp, char \* preset, int use\_jog )

move using a preset value

##### 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 2635 of file lspmac.c.

```

{
double pos;
int err;

if( preset == NULL || *preset == 0)
    return;

err = lsredis_find_preset( mp->name, preset, &pos);

if( err != 0)
    lspmac_move_or_jog_abs_queue( mp, pos, use_jog)
    ;
}

```

#### 7.6.4.43 void lspmac\_move\_preset\_queue ( lspmac\_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>	lspmac motor pointer
	<i>name</i>	Name of the preset to use

Definition at line 2284 of file lspmac.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;

mp->moveAbs( mp, pos);
lslogging_log_message( "lspmac_move_preset_queue: moving
    %s to preset '%s' (%f)", mp->name, preset_name, pos);
}

```

#### 7.6.4.44 void lspmac\_moveabs\_blight\_factor\_queue ( lspmac\_motor\_t \* mp, double pos )

Definition at line 2464 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));
}
}

```

#### 7.6.4.45 void 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 2350 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;

mp->not_done = 1;
mp->motion_seen = 0;
lspmac_SockSendDPline( mp->name, mp->write_fmt
    , mp->requested_pos_cnts);

pthread_mutex_unlock( &(mp->mutex));
}

```

#### 7.6.4.46 void lspmac\_moveabs\_flight\_factor\_queue ( lspmac\_motor\_t \* mp, double pos )

Definition at line 2443 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( &(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));
}
}

```

```

}
}

```

#### 7.6.4.47 void lspmac\_moveabs\_frontlight\_queue ( lspmac\_motor\_t \* mp, double pos )

"move" frontlight on/off

Definition at line 2431 of file lspmac.c.

```

{
pthread_mutex_lock( &(mp->mutex));
*mp->actual_pos_cnts_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));
}
}
}

```

#### 7.6.4.48 void 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 2323 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_cnts = requested_position;
if( requested_position != 0) {
    //
    // ScanEnable=0, ManualEnable=1, ManualOn=1
    //
    lspmac_SockSendDPline( mp->name, "M1124=0 M1125=1
        M1126=1");
} else {
    //
    // ManualOn=0, ManualEnable=0, ScanEnable=1
    //
    lspmac_SockSendDPline( mp->name, "M1126=0 M1125=0
        M1124=1");
}

pthread_mutex_unlock( &(mp->mutex));
}

```

#### 7.6.4.49 void 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 2655 of file lspmac.c.

```

{
    lspmac_move_or_jog_abs_queue( mp,
        requested_position, 0);
}

```

#### 7.6.4.50 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 2376 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 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);

    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
        ;
    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));
}

```

#### 7.6.4.51 void lspmac\_moveabs.wait ( lspmac\_motor\_t \* mp )

Wait for motor to finish moving.

Assume motion already queued, now just wait

##### Parameters

in	<i>mp</i>	The motor object to wait for
----	-----------	------------------------------

Definition at line 2677 of file lspmac.c.

```

    {
        struct timespec wt;
        int return_code;

        //
        // Copy the queue item for the most recent move request
        //
        pthread_mutex_lock( &(mp->mutex));

        while( mp->command_sent == 0)
            pthread_cond_wait( &mp->cond, &mp->mutex);
        pthread_mutex_unlock( &(mp->mutex));

        //
        // 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.
        //
        // This sets up a one second wait
        //
        clock_gettime( CLOCK_REALTIME, &wt);
        wt.tv_sec++;

        return_code=0;

        pthread_mutex_lock( &(mp->mutex));
        while( mp->motion_seen == 0 && return_code == 0)
            return_code = pthread_cond_timedwait( &(mp->cond), &(mp->mutex), &
                wt);

        if( return_code == 0) {
            //
            // wait for the motion that we know has started to finish
            //
            while( mp->not_done)
                pthread_cond_wait( &(mp->cond), &(mp->mutex));
        }

        //
        // 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));
    }
}

```

#### 7.6.4.52 void 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 postion (look up and send y position)

Definition at line 2219 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));
}

```

#### 7.6.4.53 void 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 2251 of file lspmac.c.

```

{
double y;
int motor_num;

pthread_mutex_lock( &(mp->mutex));

motor_num = lsredis_getl( mp->motor_num);

mp->requested_position = requested_position;

if( mp->nlut > 0 && mp->lut != NULL) {
y = lspmac_lut( mp->nlut, mp->lut, requested_position);

mp->requested_pos_cnts = (int)y;
mp->not_done = 1;
mp->motion_seen = 0;

lspmac_SockSendDPline( mp->name, "##d j=%d",
motor_num, mp->requested_pos_cnts);
}

pthread_mutex_unlock( &(mp->mutex));
}

```

#### 7.6.4.54 void lspmac\_next\_state ( )

State machine logic.

Given the current state, generate the next one

Definition at line 2050 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;
}
}

```

#### 7.6.4.55 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 1347 of file lspmac.c.

```

{
char s[512], *sp;
int homing1, homing2;
double u2c;
int motor_num;
char *fmt;
int status_changed;

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",
    // 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 & 0x000001) != (*mp->status2_p & 0x000001))
{
    lsevents_send_event( "%s %s", mp->name, (*mp->
        status2_p & 0x000001) ? "In Position" : "Moving");
}

// Get some values we might need later
//
u2c      = lsredis_getd( mp->u2c);
motor_num = lsredis_getl( mp->motor_num);

//
// 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 = lspmac_status_time
            .tv_sec - secs;
        omega_zero_time.tv_nsec= lspmac_status_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("lspmac_pmacmotor_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 & 0x000001) {
    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 & 0x000400) {
    if( mp->motion_seen == 0) {
        mp->motion_seen = 1;
        pthread_cond_signal( &(mp->cond));
    }
}

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, " ");

if( mp->nlut > 0 && mp->lut != NULL) {
    mp->position = lspmac_rlut( mp->nlut, mp->lut, mp
->actual_pos_cnts);
} else {
    if( u2c != 0.0) {
        mp->position = mp->actual_pos_cnts / u2c;
    } 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);

// set flag if we are not homed
homing1 = 0;
//          ~(homed 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;

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 & 0x000800)
    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( &(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;
}

```

#### 7.6.4.56 `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 know to what command a reply is referring. Returns the item.

Definition at line 650 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;
}

```

#### 7.6.4.57 `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 670 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;
    }

```

#### 7.6.4.58 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 626 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;
    }

```

#### 7.6.4.59 void lspmac\_Reset ( )

Clear the queue and put the PMAC into a known state.

Definition at line 749 of file lspmac.c.

```

    {
        ls_pmac_state = LS_PMAC_STATE_IDLE;

        // clear queue
        ethCmdReply = ethCmdOn;
        ethCmdOff = ethCmdOn;

        lspmac_SockFlush();
    }

```

#### 7.6.4.60 void lspmac\_reset\_queue ( )

Clear the queue as part of PMAC reinitialization.

Definition at line 613 of file lspmac.c.

```

    {
        pthread_mutex_lock( &pmac_queue_mutex);
        ethCmdOn = 0;
        ethCmdOff = 0;
        ethCmdReply = 0;
        pthread_mutex_unlock( &pmac_queue_mutex);
    }

```

#### 7.6.4.61 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 434 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;
}

```

#### 7.6.4.62 void lspmac\_run ( )

Start up the lspmac thread.

Definition at line 3263 of file lspmac.c.

```

    {
char **inits;
lspmac_motor_t *mp;
char evts[64];
int i;
int active;

pthread_create( &pmac_thread, NULL, lspmac_worker,
    NULL);

lsevents_add_listener( "CryoSwitchChanged",
    lspmac_cryoSwitchChanged_cb);
lsevents_add_listener( "scint In Position",
    lspmac_scint_inPosition_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 In Position",
    lspmac_light_zoom_cb);

for( i=0; i<lspmac_nmotors; i++) {
    snprintf( evts, sizeof( evts)-1, "%s command done", 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( NULL, '\x18')
    ;
{
    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);
}

//
// Initialize the MD2 pmac (ie, turn on the right plcc's etc)
//
/*
for( inits = lsredis_get_string_array(lspmac_md2_init); *inits != NULL;
    inits++) {
    lspmac_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<lspmac_nmotors; i++) {
    mp = &(lspmac_motors[i]);
    active = lsredis_getb( mp->active);

    // 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( "lspmac_run: motor %s is
            neither active nor inactive (!?)", mp->name);
        inits = NULL;
    }
    if( inits != NULL) {
        while( *inits != NULL) {
            lspmac_SockSendDPLine( NULL, *inits);
            inits++;
        }
    }
}
}
}

```

#### 7.6.4.63 void lspmac\_scint\_dried.cb ( char \* event )

Turn off the dryer.

##### Parameters

<i>event</i>	required by protocol
--------------	----------------------

Definition at line 3103 of file lspmac.c.

```

{
    lslogging_log_message( "lspmac_scint_dried.cb: Stopping
        dryer");
    dryer->moveAbs( dryer, 0.0);
}

```

#### 7.6.4.64 void lspmac\_scint\_inPosition.cb ( char \* event )

Maybe start drying off the scintillator.

##### Parameters

<i>event</i>	required by protocol
--------------	----------------------

Definition at line 3041 of file lspmac.c.

```

{
    double pos;
    double cover;
    int err;

    pthread_mutex_lock( &(amp;scint->mutex));
    pos = scint->position;
    err = lsredis_find_preset( scint->name, "Cover",
        &cover);
    pthread_mutex_unlock( &(amp;scint->mutex));

    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_add_timer( "scintDried", 1, 120, 0);
    }
}

```



**7.6.4.65** `pmac_cmd_queue_t* lspmac_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 688 of file lspmac.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 lspmac_push_queue( &cmd);
}

```

**7.6.4.66** `void lspmac_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 2029 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);
    }

```

#### 7.6.4.67 void lspmac\_sendcmd\_nocb ( 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 2010 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);
    }

```

#### 7.6.4.68 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 1013 of file lspmac.c.

```

    {
        pthread_mutex_lock( &ncurses_mutex);
        wprintw( term_output, "control-%c: ", '@'+ ntohs(cmd->pcmd.
                    wValue));
        pthread_mutex_unlock( &ncurses_mutex);
        hex_dump( nreceived, (unsigned char *)buff);
        pthread_mutex_lock( &ncurses_mutex);
        wnoutrefresh( term_output);
        wnoutrefresh( term_input);
        doupdate();
        pthread_mutex_unlock( &ncurses_mutex);
    }

```

**7.6.4.69 void lspmac\_Service ( struct pollfd \* evt )**

Service routine for packet coming from the PMAC.

All communications is asynchronous so this is the only place incoming packets are handled

**Parameters**

in	evt	pollfd object returned by poll
----	-----	--------------------------------

Definition at line 796 of file lspmac.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 = lspmac_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_PMAC_GETMEM)
            ls_pmac_state = LS_PMAC_STATE_GMR;
        else if( cmd->no_reply == 0)
            ls_pmac_state = LS_PMAC_STATE_WACK;
        else
            ls_pmac_state = LS_PMAC_STATE_WACK_NFR
        ;
        break;

case LS_PMAC_STATE_CR:
        nsent = send( evt->fd, &cr_cmd, pmac_cmd_size, 0);
        gettimeofday( &pmac_time_sent, NULL);
        ls_pmac_state = LS_PMAC_STATE_WCR;
        break;

```

```

case LS_PMAC_STATE_RR:
    nsent = send( evt->fd, &rr_cmd, pmac_cmd_size, 0);
    gettimeofday( &pmac_time_sent, NULL);
    ls_pmac_state = LS_PMAC_STATE_WACK_RR;
    break;

case LS_PMAC_STATE_GB:
    nsent = send( evt->fd, &gb_cmd, pmac_cmd_size, 0);
    gettimeofday( &pmac_time_sent, NULL);
    ls_pmac_state = LS_PMAC_STATE_WGB;
    break;
}
}

if( evt->revents & POLLIN) {

    if( receiveBufferSize - receiveBufferIn < 1400) {
        char *newbuff;

        receiveBufferSize += 1400;
        newbuff = calloc( receiveBufferSize, sizeof( unsigned char));
        if( newbuff == NULL) {
            lslogging_log_message( "lspmac_Service: Out of
memory");
            exit( -1);
        }
        if( receiveBuffer != NULL) {
            memcpy( newbuff, receiveBuffer, receiveBufferIn);
            free(receiveBuffer);
        }
        receiveBuffer = newbuff;
    }

    nread = read( evt->fd, receiveBuffer + receiveBufferIn, 1400);

    foundEOCR = 0;
    if( ls_pmac_state == LS_PMAC_STATE_GMR) {
        //
        // get memory returns binary stuff, don't try to parse it
        //
        receiveBufferIn += nread;
    } else {
        //
        // other commands end in 6 if OK, 7 if not
        //
        for( i=receiveBufferIn; i<receiveBufferIn+nread; i++) {
            if( receiveBuffer[i] == 7) {
                //
                // Error condition
                //
                lspmac_Error( &(receiveBuffer[i]));
                receiveBufferIn = 0;
                return;
            }
            if( receiveBuffer[i] == 6) {
                //
                // End of command response
                //
                foundEOCR = 1;
                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 = lspmac_pop_reply();
        ls_pmac_state = LS_PMAC_STATE_IDLE;
        break;

    case LS_PMAC_STATE_WCR:
        cmd = lspmac_pop_reply();
        ls_pmac_state = LS_PMAC_STATE_IDLE;
        break;
    case LS_PMAC_STATE_WGB:
        if( foundEOCR) {
            cmd = lspmac_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;
    }
}
}
}

```

#### 7.6.4.70 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.

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 1169 of file lspmac.c.

```

{
    //
    // track the shutter state and signal if it has changed
    //
    pthread_mutex_lock( &lspmac_shutter_mutex);
    if( md2_status.fs_has_opened && !
        lspmac_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( &lspmac_shutter_cond);
    }
    lspmac_shutter_has_opened = md2_status.
        fs_has_opened;

    if( lspmac_shutter_state != md2_status.
        fs_is_open) {
        lspmac_shutter_state = md2_status.fs_is_open
        ;
        pthread_cond_signal( &lspmac_shutter_cond);
    }

    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( &lspmac_shutter_mutex);
}

```

#### 7.6.4.71 void lspmac\_SockFlush ( )

Reset the PMAC socket from the PMAC side.

Puts the PMAC into a known communications state

Definition at line 742 of file lspmac.c.

```
{
    lspmac_send_command( VR_DOWNLOAD, VR_PMAC_FLUSH
        , 0, 0, 0, NULL, NULL, 1, NULL);
}
```

#### 7.6.4.72 pmac\_cmd\_queue\_t\* lspmac\_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 1048 of file lspmac.c.

```
{
    return lspmac_send_command( VR_UPLOAD,
        VR_PMAC_GETMEM, offset, 0, nbytes, NULL, lspmac_GetmemReplyCB
        , 0, NULL);
}
```

#### 7.6.4.73 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 1101 of file lspmac.c.

```
{
    return lspmac_send_command( VR_DOWNLOAD,
        VR_PMAC_SENDCTRLCHAR, c, 0, 0, NULL,
        lspmac_SendControlReplyPrintCB, 0, event);
}
```

#### 7.6.4.74 void lspmac\_SockSendDPLine ( char \* event, char \* fmt, ... )

prepare (queue up) a line to send the dpram ascii command interface

Definition at line 1910 of file lspmac.c.

```
{
    va_list arg_ptr;
    uint32_t index;
    char *pl;

    pthread_mutex_lock( &lspmac_ascii_mutex);
    index = lspmac_dpascii_on++ % LSPMAC_DPASCII_QUEUE_LENGTH
;
}
```

```

    pl = lspmac_dpascii_queue[index].pl;

    va_start( arg_ptr, fmt);
    vsnprintf( pl, 159, fmt, arg_ptr);
    pl[159] = 0;
    va_end( arg_ptr);

    lspmac_dpascii_queue[index].event = event;

    pthread_mutex_unlock( &lspmac_ascii_mutex);
}

```

#### 7.6.4.75 void lspmac\_SockSendDPqueue ( )

Definition at line 1930 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
        ->pl);

    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->pl)+1, qp->pl, 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);
}

```

#### 7.6.4.76 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 1058 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( payload);

    return lspmac_send_command( VR_DOWNLOAD,

```

```

    VR_PMAC_SENDLINE, 0, 0, strlen( payload), payload,
    lspmac_GetShortReplyCB, 0, event);
}

```

#### 7.6.4.77 `lspmac_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 1081 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);
}

```

#### 7.6.4.78 `lspmac_motor_t* lspmac_soft_motor_init ( lspmac_motor_t * d, char * name, void(*) (lspmac_motor_t *, double) moveAbs )`

Definition at line 2860 of file `lspmac.c`.

```

{
    _lspmac_motor_init( d, name);

    d->moveAbs = moveAbs;
    d->read = lspmac_soft_motor_read;
    d->actual_pos_cnts_p = calloc( sizeof(int), 1);
    *d->actual_pos_cnts_p = 0;

    return d;
}

```

#### 7.6.4.79 `void lspmac_soft_motor_read ( lspmac_motor_t * p )`

Dummy routine to read a soft motor.

Definition at line 2855 of file `lspmac.c`.

```

{
}

```

#### 7.6.4.80 `int lspmac_test_preset ( lspmac_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 2302 of file `lspmac.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 - lspmac_getPosition( mp)) <=
            tolerance)
            return 1;

        return 0;
    }

```

#### 7.6.4.81 void lspmac\_video\_rotate( double secs )

Special motion program to collect centering video.

Definition at line 2486 of file lspmac.c.

```

    {
        double q10;           // starting position (counts)
        double q11;           // delta counts
        double q12;           // milliseconds to run over delta

        double u2c;

        if( secs <= 0.0)
            return;

        omega_zero_search = 1;

        pthread_mutex_lock( &(omega->mutex));
        u2c = lsredis_getd( omega->u2c);

        q10 = 0;
        q11 = 360.0 * u2c;
        q12 = 1000 * secs;

        omega_zero_velocity = 360.0 * u2c / secs; //
            counts/second to back calculate zero crossing time

        lspmac_SockSendDPLine( omega->name, "&1
            Q10=%.1f Q11=%.1f Q12=%.1f Q13=(I117) Q14=(I116) B240R", q10, q11, q12);
        pthread_mutex_unlock( &(omega->mutex));
    }

```

#### 7.6.4.82 void\* lspmac\_worker( void \* dummy )

Our lspmac worker thread.

##### Parameters

in	<i>dummy</i>	Unused but required by pthread library
----	--------------	--

Definition at line 2168 of file lspmac.c.

```

    {
        static int disconnected_notify = 0;
        static int old_state;

        old_state = ls_pmac_state;
        while( 1) {
            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);
}
}
}

```

#### 7.6.4.83 void lspmac\_zoom\_lut\_setup ( )

Set up lookup table for zoom.

Definition at line 3111 of file lspmac.c.

```

{
    int i;
    lsredis_obj_t *p;

    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);
    }
    pthread_mutex_unlock( &zoom->mutex);
}

```

### 7.6.5 Variable Documentation

#### 7.6.5.1 lspmac\_motor\_t\* alignx

Alignment stage X.

Definition at line 88 of file Ispmac.c.

#### 7.6.5.2 Ispmac\_motor\_t\* aligny

Alignment stage Y.

Definition at line 89 of file Ispmac.c.

#### 7.6.5.3 Ispmac\_motor\_t\* alignz

Alignment stage X.

Definition at line 90 of file Ispmac.c.

#### 7.6.5.4 Ispmac\_motor\_t\* anal

Polaroid analyzer motor.

Definition at line 91 of file Ispmac.c.

#### 7.6.5.5 Ispmac\_motor\_t\* apery

Aperture Y.

Definition at line 93 of file Ispmac.c.

#### 7.6.5.6 Ispmac\_motor\_t\* aperz

Aperture Z.

Definition at line 94 of file Ispmac.c.

#### 7.6.5.7 Ispmac\_bi\_t\* arm\_parked

(whose arm? parked where?)

Definition at line 131 of file Ispmac.c.

#### 7.6.5.8 Ispmac\_motor\_t\* blight

Back Light DAC.

Definition at line 105 of file Ispmac.c.

#### 7.6.5.9 Ispmac\_bi\_t\* blight\_down

Backlight is down.

Definition at line 121 of file Ispmac.c.

#### 7.6.5.10 Ispmac\_motor\_t\* blight\_f

Back light scale factor.

Definition at line 114 of file Ispmac.c.

#### 7.6.5.11 `lspmac_motor_t* blight_ud`

Back light Up/Down actuator.

Definition at line 109 of file `lspmac.c`.

#### 7.6.5.12 `lspmac_bi_t* blight_up`

Backlight is up.

Definition at line 122 of file `lspmac.c`.

#### 7.6.5.13 `lspmac_motor_t* capy`

Capillary Y.

Definition at line 95 of file `lspmac.c`.

#### 7.6.5.14 `lspmac_motor_t* capz`

Capillary Z.

Definition at line 96 of file `lspmac.c`.

#### 7.6.5.15 `lspmac_motor_t* cenx`

Centering Table X.

Definition at line 98 of file `lspmac.c`.

#### 7.6.5.16 `lspmac_motor_t* ceny`

Centering Table Y.

Definition at line 99 of file `lspmac.c`.

#### 7.6.5.17 `pmac_cmd_t cr_cmd` `[static]`

commands to send out "readready", "getbuffer", "controlresponse" (initialized in main)

Definition at line 183 of file `lspmac.c`.

#### 7.6.5.18 `lspmac_motor_t* cryo`

Move the cryostream towards or away from the crystal.

Definition at line 110 of file `lspmac.c`.

#### 7.6.5.19 `lspmac_bi_t* cryo_back`

cryo is in the back position

Definition at line 123 of file `lspmac.c`.

**7.6.5.20 lspmac\_bi\_t\* cryo\_switch**

that little toggle switch for the cryo

Definition at line 120 of file lspmac.c.

**7.6.5.21 unsigned char dbmem[64 \* 1024] [static]**

double buffered memory

Definition at line 172 of file lspmac.c.

**7.6.5.22 int dbmemIn = 0 [static]**

next location

Definition at line 173 of file lspmac.c.

**7.6.5.23 lspmac\_motor\_t\* dryer**

blow air on the scintillator to dry it off

Definition at line 111 of file lspmac.c.

**7.6.5.24 lspmac\_bi\_t\* etel\_init\_ok**

ETEL initialized OK.

Definition at line 128 of file lspmac.c.

**7.6.5.25 lspmac\_bi\_t\* etel\_on**

ETEL is on.

Definition at line 127 of file lspmac.c.

**7.6.5.26 lspmac\_bi\_t\* etel\_ready**

ETEL is ready.

Definition at line 126 of file lspmac.c.

**7.6.5.27 unsigned int ethCmdOff = 0 [static]**

points to current command (or none if == ethCmdOn)

Definition at line 186 of file lspmac.c.

**7.6.5.28 unsigned int ethCmdOn = 0 [static]**

points to next empty PMAC command queue position

Definition at line 185 of file lspmac.c.

**7.6.5.29 pmac\_cmd\_queue\_t ethCmdQueue[PMAC\_CMD\_QUEUE\_LENGTH] [static]**

PMAC command queue.

Definition at line 184 of file lspmac.c.

**7.6.5.30 unsigned int ethCmdReply = 0 [static]**

Used like ethCmdOff only to deal with the pmac reply to a command.

Definition at line 187 of file lspmac.c.

**7.6.5.31 lspmac\_motor\_t\* flight**

Front Light DAC.

Definition at line 104 of file lspmac.c.

**7.6.5.32 lspmac\_motor\_t\* flight\_f**

Front light scale factor.

Definition at line 115 of file lspmac.c.

**7.6.5.33 lspmac\_motor\_t\* flight\_oo**

Turn front light on/off.

Definition at line 113 of file lspmac.c.

**7.6.5.34 lspmac\_motor\_t\* fluo**

Move the fluorescence detector in/out.

Definition at line 112 of file lspmac.c.

**7.6.5.35 lspmac\_bi\_t\* fluor\_back**

fluor is in the back position

Definition at line 124 of file lspmac.c.

**7.6.5.36 lspmac\_motor\_t\* fscint**

Scintillator Piezo DAC.

Definition at line 106 of file lspmac.c.

**7.6.5.37 lspmac\_motor\_t\* fshut**

Fast shutter.

Definition at line 103 of file lspmac.c.

**7.6.5.38** `pmac_cmd_t gb_cmd` `[static]`

Definition at line 183 of file `lspmac.c`.

**7.6.5.39** `int getivars = 0` `[static]`

flag set at initialization to send i vars to db

Definition at line 79 of file `lspmac.c`.

**7.6.5.40** `int getmvars = 0` `[static]`

flag set at initialization to send m vars to db

Definition at line 80 of file `lspmac.c`.

**7.6.5.41** `lspmac_bi_t* hp_air`

High pressure air OK.

Definition at line 119 of file `lspmac.c`.

**7.6.5.42** `lspmac_motor_t* kappa`

Kappa.

Definition at line 100 of file `lspmac.c`.

**7.6.5.43** `lspmac_bi_t* lp_air`

Low pressure air OK.

Definition at line 118 of file `lspmac.c`.

**7.6.5.44** `int ls_pmac_state = LS_PMAC_STATE_DETACHED` `[static]`

Current state of the PMAC communications state machine.

Definition at line 51 of file `lspmac.c`.

**7.6.5.45** `lspmac_ascii_buffers_t lspmac_ascii_buffers` `[static]`

Definition at line 352 of file `lspmac.c`.

**7.6.5.46** `pthread_mutex_t lspmac_ascii_buffers_mutex`

Definition at line 353 of file `lspmac.c`.

**7.6.5.47** `int lspmac_ascii_busy = 0` `[static]`

flag for condition to wait for

Definition at line 66 of file `lspmac.c`.

**7.6.5.48** `pthread_mutex_t lspmac_ascii_mutex` `[static]`

Keep too many processes from sending commands at once.

Definition at line 65 of file `lspmac.c`.

**7.6.5.49** `lspmac_bi_t lspmac_bis[32]`

array of binary inputs

Definition at line 82 of file `lspmac.c`.

**7.6.5.50** `uint32_t lspmac_dpascii_off = 0` `[static]`

Definition at line 363 of file `lspmac.c`.

**7.6.5.51** `uint32_t lspmac_dpascii_on = 0` `[static]`

Definition at line 362 of file `lspmac.c`.

**7.6.5.52** `lspmac_dpascii_queue_t lspmac_dpascii_queue[LSPMAC_DPASCII_QUEUE_LENGTH]` `[static]`

Definition at line 361 of file `lspmac.c`.

**7.6.5.53** `lsredis_obj_t* lspmac_md2_init` `[static]`

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

**7.6.5.54** `lspmac_motor_t lspmac_motors[48]`

All our motors.

Definition at line 85 of file `lspmac.c`.

**7.6.5.55** `pthread_cond_t lspmac_moving_cond`

Wait for motor(s) to finish moving condition.

Definition at line 62 of file `lspmac.c`.

**7.6.5.56** `int lspmac_moving_flags`

Flag used to implement motor moving condition.

Definition at line 63 of file `lspmac.c`.

**7.6.5.57** `pthread_mutex_t lspmac_moving_mutex`

Coordinate moving motors between threads.

Definition at line 61 of file `lspmac.c`.



**7.6.5.58 int lspmac\_nbis = 0**

number of active binary inputs

Definition at line 83 of file lspmac.c.

**7.6.5.59 int lspmac\_nmotors = 0**

The number of motors we manage.

Definition at line 86 of file lspmac.c.

**7.6.5.60 pthread\_cond\_t lspmac\_shutter\_cond**

Allows waiting for the shutter status to change.

Definition at line 60 of file lspmac.c.

**7.6.5.61 int lspmac\_shutter\_has\_opened**

Indicates that the shutter had opened, perhaps briefly even if the state did not change.

Definition at line 58 of file lspmac.c.

**7.6.5.62 pthread\_mutex\_t lspmac\_shutter\_mutex**

Coordinates threads reading shutter status.

Definition at line 59 of file lspmac.c.

**7.6.5.63 int lspmac\_shutter\_state**

State of the shutter, used to detect changes.

Definition at line 57 of file lspmac.c.

**7.6.5.64 struct timespec lspmac\_status\_last\_time [static]**

Time the status was read.

Definition at line 72 of file lspmac.c.

**7.6.5.65 struct timespec lspmac\_status\_time [static]**

Time the status was read.

Definition at line 71 of file lspmac.c.

**7.6.5.66 md2\_status\_t md2\_status [static]**

Buffer for MD2 Status.

Definition at line 338 of file lspmac.c.

**7.6.5.67 pthread\_mutex\_t md2\_status\_mutex**

Synchronize reading/writing status buffer.

Definition at line 339 of file lspmac.c.

**7.6.5.68 lspmac\_bi\_t\* minikappa\_ok**

Minikappa is OK (whatever that means)

Definition at line 129 of file lspmac.c.

**7.6.5.69 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 179 of file lspmac.c.

**7.6.5.70 lspmac\_motor\_t\* omega**

MD2 omega axis (the air bearing)

Definition at line 87 of file lspmac.c.

**7.6.5.71 int omega\_zero\_search = 0 [static]**

Indicate we'd really like to know when omega crosses zero.

Definition at line 68 of file lspmac.c.

**7.6.5.72 struct timespec omega\_zero\_time**

Time we believe that omega crossed zero.

Definition at line 70 of file lspmac.c.

**7.6.5.73 double omega\_zero\_velocity = 0 [static]**

rate (cnts/sec) that omega was traveling when it crossed zero

Definition at line 69 of file lspmac.c.

**7.6.5.74 lspmac\_motor\_t\* phi**

Phi (not data collection axis)

Definition at line 101 of file lspmac.c.

**7.6.5.75 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 auziliary 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-chage 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 190 of file lspmac.c.

#### 7.6.5.76 pthread\_cond\_t pmac\_queue\_cond

wait for a command to be sent to PMAC before continuing

Definition at line 76 of file lspmac.c.

#### 7.6.5.77 pthread\_mutex\_t pmac\_queue\_mutex

manage access to the pmac command queue

Definition at line 75 of file lspmac.c.

#### 7.6.5.78 pthread\_t pmac\_thread [static]

our thread to manage access and communication to the pmac

Definition at line 74 of file lspmac.c.

#### 7.6.5.79 struct pollfd pmacfd [static]

our poll structure

Definition at line 77 of file lspmac.c.

#### 7.6.5.80 pmac\_cmd\_t rr\_cmd [static]

Definition at line 183 of file lspmac.c.

#### 7.6.5.81 lspmac\_bi\_t\* sample\_detected

smart magnet detected sample

Definition at line 125 of file lspmac.c.

#### 7.6.5.82 lspmac\_motor\_t\* scint

Scintillator Z.

Definition at line 97 of file lspmac.c.

#### 7.6.5.83 `lspmac_bi_t* shutter_open`

shutter is open (note in pmc says this is a slow input)

Definition at line 132 of file lspmac.c.

#### 7.6.5.84 `lspmac_bi_t* smart_mag_err`

smart magnet error (coil broken perhaps)

Definition at line 133 of file lspmac.c.

#### 7.6.5.85 `lspmac_bi_t* smart_mag_off`

smart magnet is off

Definition at line 134 of file lspmac.c.

#### 7.6.5.86 `lspmac_bi_t* smart_mag_on`

smart magnet is on

Definition at line 130 of file lspmac.c.

#### 7.6.5.87 `lspmac_motor_t* smart_mag_oo`

Smart Magnet on/off.

Definition at line 108 of file lspmac.c.

#### 7.6.5.88 `lspmac_motor_t* zoom`

Optical zoom.

Definition at line 92 of file lspmac.c.

## 7.7 `lsredis.c` File Reference

Support redis hash synchronization.

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

### 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 function 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\\_regexec](#) (const regex\_t \*preg, [lsredis\\_obj\\_t](#) \*p, size\_t nmatch, regmatch\_t \*pmatch, int eflags)
- char \* [lsredis\\_getstr](#) ([lsredis\\_obj\\_t](#) \*p)
  - return a copy of the key's string value*
- void [lsredis\\_setstr](#) ([lsredis\\_obj\\_t](#) \*p, char \*fmt,...)
  - Set the value and update redis.*
- double [lsredis\\_getd](#) ([lsredis\\_obj\\_t](#) \*p)
- long int [lsredis\\_getl](#) ([lsredis\\_obj\\_t](#) \*p)
- 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.*
- int [lsredis\\_find\\_preset](#) (char \*base, char \*preset\_name, double \*dval)
- void [lsredis\\_init](#) (char \*pub, char \*re, char \*head)
  - 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](#)
- static pthread\_mutex\_t [lsredis\\_mutex](#)
- static pthread\_cond\_t [lsredis\\_cond](#)
- static int [lsredis\\_running](#) = 0
- 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 struct pollfd [subfd](#)
- static struct pollfd [rofd](#)
- static struct pollfd [wrfd](#)

### 7.7.1 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 list 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
```

(this last value is used to support the now depreciated `px.kvs` table in the LS-CAT postgresql server). We assume that all publisher 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	<code>*lsredis_getstr( lsredis_obj_t *p)</code>	Returns a copy of the VALUE field. Use
double	<code>lsredis_getd( lsredis_obj_t *p)</code>	Returns a double. If the value was not
long int	<code>lsredis_getl( lsredis_obj_t *p)</code>	Returns a long int. If the value was not
char	<code>**lsredis_get_string_array( lsredis_obj_t *p)</code>	Returns an array of string pointers. V or NULL if the value could not be parsed
int	<code>lsredis_getb( lsredis_obj_t *p)</code>	Returns 1, 0, or -1 based on the first c
char	<code>lsredis_getc( lsredis_obj_t *p)</code>	Returns the first character of VALUE

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

## 7.7.2 Function Documentation

### 7.7.2.1 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 437 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, NULL);
    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;
}

```

### 7.7.2.2 void lsredis\_set\_value( lsredis\_obj\_t \*p, char \*v )

set\_value and setstr helper function p->mutex must be locked before calling

Definition at line 145 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 = lspg_array2ptrs( p->value);

    switch( *(p->value)) {
        case 'T':

```



```

    case 't':
    case 'Y':
    case 'y':
    case 'l':
        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
    );
    }
}

```

#### 7.7.2.3 void Isredis\_addRead ( void \* data )

hook to mange read events

Definition at line 567 of file Isredis.c.

```

{
    struct pollfd *pfd;
    pfd = (struct pollfd *)data;

    if( (pfd->events & POLLIN) == 0 ) {
        pfd->events |= POLLIN;
        pthread_kill( lsredis_thread, SIGUSR1);
    }
}

```

#### 7.7.2.4 void Isredis\_addWrite ( void \* data )

hook to manage write events

Definition at line 591 of file Isredis.c.

```

{
    struct pollfd *pfd;
    pfd = (struct pollfd *)data;

    if( (pfd->events & POLLOUT) == 0 ) {
        pfd->events |= POLLOUT;
        pthread_kill( lsredis_thread, SIGUSR1);
    }
}

```

#### 7.7.2.5 void Isredis\_cleanup ( void \* data )

hook to clean up TODO: figure out what we are supposed to do here and do it

Definition at line 616 of file Isredis.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);
}
}

```

#### 7.7.2.6 int lsredis\_cmpnstr ( lsredis\_obj\_t \* p, char \* s, int n )

Definition at line 235 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;
}

```

#### 7.7.2.7 int lsredis\_cmpstr ( lsredis\_obj\_t \* p, char \* s )

Definition at line 224 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;
}

```

#### 7.7.2.8 void lsredis\_debugCB ( redisAsyncContext \* ac, void \* reply, void \* privdata )

Log the reply.

Definition at line 95 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);
}
}

```

#### 7.7.2.9 void Lsredis.delRead ( void \* data )

hook to manage "don't need to read" events

Definition at line 579 of file Lsredis.c.

```

{
    struct pollfd *pfd;
    pfd = (struct pollfd *)data;

    if( (pfd->events & POLLIN) != 0) {
        pfd->events &= ~POLLIN;
        pthread_kill( lsredis_thread, SIGUSR1);
    }
}

```

#### 7.7.2.10 void Lsredis.delWrite ( void \* data )

hook to manage "don't need to write anymore" events

Definition at line 603 of file Lsredis.c.

```

{
    struct pollfd *pfd;
    pfd = (struct pollfd *)data;

    if( (pfd->events & POLLOUT) != 0) {
        pfd->events &= ~POLLOUT;
        pthread_kill( lsredis_thread, SIGUSR1);
    }
}

```

#### 7.7.2.11 void Lsredis.fd\_service ( struct pollfd \* evt )

service the socket requests

Definition at line 888 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);
}

```

### 7.7.2.12 int lsredis\_find\_preset( char \*base, char \*preset\_name, double \*dval )

Definition at line 755 of file lsredis.c.

```

{
char s[512];
int i;
int err;
ENTRY htab_input, *htab_output;
lsredis_obj_t *p;

i = 0;
for( i=0; i<1024; i++) {
    snprintf( s, sizeof( s)-1, "%s.%s.preset.%d.name", lsredis_head
, base, i);
    s[sizeof(s)-1] = 0;
    htab_input.key = s;
    htab_input.data = NULL;
    err = hsearch_r( htab_input, FIND, &htab_output, &lsredis_htab
);
    if( err == 0) {
        // We've run out of names to look for: done
        lslogging_log_message( "lsredis_find_preset: no
        preset for motor %s named '%s'", base, preset_name);
        *dval = 0.0;
        return 0;
    }

    // Check if we have a match
    p = htab_output->data;
    if( lsredis_cmpstr( p, preset_name)==0) {
        // got a match, now look for the position
        snprintf( s, sizeof( s)-1, "%s.%s.preset.%d.position", lsredis_head
, base, i);
        s[sizeof(s)-1] = 0;
        htab_input.key = s;
        htab_input.data = NULL;
        err = hsearch_r( htab_input, FIND, &htab_output, &lsredis_htab
);
        if( err == 0) {
            // Name but not position? odd.
            lslogging_log_message( "lsredis_find_preset:
            Error, motor %s preset '%s' has no position defined", base, preset_name);
            *dval = 0.0;
            return 0;
        }
        p = htab_output->data;
        *dval = lsredis_getd( p);
        return 1;
    }
}
// How'd we get here?
// did someone really define that many presets? And then looked for one
// that's not there?
*dval = 0;
return 0;
}

```

### 7.7.2.13 lsredis\_obj\_t\* lsredis\_get\_obj( char \*fmt, ... )

Definition at line 523 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;
}

```

#### 7.7.2.14 char\*\* lsredis\_get\_string\_array( lsredis\_obj\_t \* p )

Definition at line 364 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;
}

```

#### 7.7.2.15 int lsredis\_getb( lsredis\_obj\_t \* p )

Definition at line 377 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;
}

```

#### 7.7.2.16 char lsredis\_getc( lsredis\_obj\_t \* p )

Definition at line 390 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;
    }

```

#### 7.7.2.17 double lsredis\_getd( lsredis\_obj\_t \* p )

Definition at line 338 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;
    }

```

#### 7.7.2.18 long int lsredis\_getl( lsredis\_obj\_t \* p )

Definition at line 351 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;
    }

```

#### 7.7.2.19 char\* lsredis\_getstr( lsredis\_obj\_t \* p )

return a copy of the key's string value

Definition at line 262 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;
    }

```

**7.7.2.20 void lsredis\_hgetCB ( redisAsyncContext \* *ac*, void \* *reply*, void \* *privdata* )**

Definition at line 403 of file Isredis.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);
    }
}

```

**7.7.2.21 void lsredis\_init ( char \* *pub*, char \* *re*, char \* *head* )**

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 809 of file Isredis.c.

```

{
    int err;
    int nerrmsg;
    char *errmsg;

    //
    // set up hash map to store redis objects
    //
    err = hcreate_r( 8192, &lsredis_hstab);
    if( err == 0) {
        lslogging_log_message( "lsredis_init: Cannot create
            hash table. Really bad things are going to happen. hcreate_r returned %d", err);
    }

    lsredis_head = strdup( head);
    lsredis_publisher = strdup( pub);

    pthread_mutex_init( &lsredis_mutex, NULL);
    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("10.1.0.3", 6379);
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;

err = regcomp( &lsredis_key_select_regex, re,
    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_select: %s", errmsg);
        free( errmsg);
    }
}
}
}

```

#### 7.7.2.22 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 734 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);
    }
}
}
}

```



**7.7.2.23 void Isredis\_maybe\_add\_key ( char \* *k* )**

Definition at line 726 of file Isredis.c.

```

{
    if( regexec( &lsredis_key_select_regex, k, 0, NULL, 0
    ) == 0) {
        _lsredis_get_obj( k);
    }
}

```

**7.7.2.24 int Isredis\_regexec ( const regex\_t \* *preg*, Isredis\_obj\_t \* *p*, size\_t *nmatch*, regmatch\_t \* *pmatch*, int *eflags* )**

Definition at line 246 of file Isredis.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;
}

```

**7.7.2.25 void Isredis\_run ( )**

Definition at line 1013 of file Isredis.c.

```

{
    pthread_create( &lsredis_thread, NULL, lsredis_worker
    , NULL);
}

```

**7.7.2.26 void Isredis\_set\_value ( Isredis\_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 206 of file Isredis.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);
}

```

### 7.7.2.27 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. Arbitrary 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 its packet (before it returns) so we should be OK with this location disappearing soon.

Definition at line 287 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);
}

```

### 7.7.2.28 void lsredis\_sig\_service ( struct pollfd \* evt )

Parameters

in	evt	The pollfd object that triggered this call
----	-----	--

Definition at line 912 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));
    }

```

#### 7.7.2.29 void lsredis\_subCB ( redisAsyncContext \* ac, void \* reply, void \* privdata )

Use the publication to request the new value.

Definition at line 634 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( regex( &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);
}
}
}

```

### 7.7.2.30 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 931 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;
}

```

```

if( redisAsyncCommand( subac, lsredis_subCB, NULL, "
    PSUBSCRIBE REDIS_KV_CONNECTOR UI* MD2-*" ) == REDIS_ERR) {
    lslogging_log_message( "Error sending PSUBSCRIBE
        command");
}

redisAsyncCommand( roac, lsredis_keysCB, NULL, "KEYS *");
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]));
        }
    }
}
}

```

#### 7.7.2.31 void redisDisconnectCB ( const redisAsyncContext \* ac, int status )

call back in case a redis server becomes disconnected TODO: reconnect

Definition at line 559 of file Lsredis.c.

```

{
if( status != REDIS_OK) {
    lslogging_log_message( "lsredis: Disconnected with
        status %d", status);
}
}

```

### 7.7.3 Variable Documentation

#### 7.7.3.1 pthread\_cond\_t lsredis\_cond [static]

Definition at line 74 of file Lsredis.c.

#### 7.7.3.2 char\* lsredis\_head = NULL [static]

Definition at line 87 of file Lsredis.c.

**7.7.3.3** `struct hsearch_data lsredis_hstab` `[static]`

Definition at line 79 of file lsredis.c.

**7.7.3.4** `regex_t lsredis_key_select_regex` `[static]`

Definition at line 86 of file lsredis.c.

**7.7.3.5** `pthread_mutex_t lsredis_mutex` `[static]`

Definition at line 73 of file lsredis.c.

**7.7.3.6** `lsredis_obj_t* lsredis_objs = NULL` `[static]`

Definition at line 78 of file lsredis.c.

**7.7.3.7** `char* lsredis_publisher = NULL` `[static]`

Definition at line 85 of file lsredis.c.

**7.7.3.8** `int lsredis_running = 0` `[static]`

Definition at line 75 of file lsredis.c.

**7.7.3.9** `pthread_t lsredis_thread` `[static]`

Definition at line 71 of file lsredis.c.

**7.7.3.10** `redisAsyncContext* roac` `[static]`

Definition at line 82 of file lsredis.c.

**7.7.3.11** `struct pollfd rofd` `[static]`

Definition at line 90 of file lsredis.c.

**7.7.3.12** `redisAsyncContext* subac` `[static]`

Definition at line 81 of file lsredis.c.

**7.7.3.13** `struct pollfd subfd` `[static]`

Definition at line 89 of file lsredis.c.

**7.7.3.14** `redisAsyncContext* wrac` `[static]`

Definition at line 83 of file lsredis.c.

## 7.7.3.15 struct pollfd wrfd [static]

Definition at line 91 of file lsredis.c.

## 7.8 ltimer.c File Reference

Support for delayed and periodic events.

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

### Data Structures

- struct [ltimer\\_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 [ltimer\\_list\\_struct](#) [ltimer\\_list\\_t](#)  
*Everything we need to know about a timer.*

### Functions

- void [ltimer\\_add\\_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 \* [ltimer\\_worker](#) (void \*dummy)  
*Our worker.*
- void [ltimer\\_init](#) ()  
*Initialize the timer list and pthread stuff.*
- void [ltimer\\_run](#) ()  
*Start up our thread.*

### Variables

- static int [ltimer\\_active\\_timers](#) = 0  
*count of the number timers we are tracking*
- static [ltimer\\_list\\_t](#) [ltimer\\_list](#) [[LSTIMER\\_LIST\\_LENGTH](#)]  
*Our timer list.*
- static pthread\_t [ltimer\\_thread](#)

- the timer thread*
- static pthread\_mutex\_t [lstimer\\_mutex](#)  
*protect the timer list*
- static pthread\_cond\_t [lstimer\\_cond](#)  
*allows us to be idle when there is nothing to do*
- static timer\_t [lstimer\\_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 [lstimer.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 [lstimer.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 [lstimer.c](#).

### 7.8.3 Typedef Documentation

#### 7.8.3.1 typedef struct lstimer\_list\_struct lstimer\_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 174 of file [lstimer.c](#).



```

    {
        pthread_mutex_lock( &lstimer_mutex);
        service_timers();
        pthread_mutex_unlock( &lstimer_mutex);
    }

```

#### 7.8.4.2 void lstimer\_add\_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 50 of file lstimer.c.

```

    {
        int i;
        struct timespec now;

        // Time we were called. Delay is based on call time, not queued time
        //
        clock_gettime( CLOCK_REALTIME, &now);

        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_add_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);
    }

```

#### 7.8.4.3 void lstimer\_init ( )

Initialize the timer list and pthread stuff.

Definition at line 259 of file lstimer.c.

```

    {
        int i;

        for( i=0; i<LSTIMER_LIST_LENGTH; i++) {
            lstimer_list[i].shots = 0;
        }

        pthread_mutex_init( &lstimer_mutex, NULL);
        pthread_cond_init( &lstimer_cond, NULL);
    }

```

#### 7.8.4.4 void lstimer\_run ( )

Start up our thread.

Definition at line 273 of file lstimer.c.

```

    {
        pthread_create( &lstimer_thread, NULL, lstimer_worker
            , NULL);
    }

```

#### 7.8.4.5 static void\* lstimer\_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 184 of file lstimer.c.

```

    {
        int
            known_timers;

        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( "lstimer_worker: sigaction
                failed");
            exit( -1);
        }

        // Create the timer
        //
        sev.sigev_notify = SIGEV_SIGNAL;
        sev.sigev_signo = SIGRTMIN;
        sev.sigev_value.sival_ptr = &lstimer_timerid;
        timer_create( CLOCK_REALTIME, &sev, &lstimer_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);

known_timers = 0;

while( 1) {
    pthread_mutex_lock( &lstmimer_mutex);

    while( new_timer == 0)
        pthread_cond_wait( &lstmimer_cond, &lstmimer_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( &lstmimer_mutex);

    // Let the signals rain down
    //
    sigprocmask( SIG_UNBLOCK, &mask, NULL);
}
}

```

#### 7.8.4.6 static void service\_timers ( ) [static]

Send events that are past due, due, or just about to be due.

Definition at line 102 of file ltimer.c.

```

{
    int
    i,
    found_active;

    lstmimer_list_t *p;
    struct timespec now, then, soonest;
    struct itimerspec its;

    //
    // Did I remind you not to let this thread own the lstmimer mutex outside of
    // this
    // service routine when SIGRTMIN is active?
    //

    // Call with lstmimer_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<lstmimer_active_timers; i++) {
        p = &(lstmimer_list[i]);
        if( p->shots != 0) {
            found_active++;
            if( p->next_secs < then.tv_sec || (p->next_secs ==
            then.tv_sec && p->next_nsec <= 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 lstimer_cond` [static]

allows us to be idle when there is nothing to do

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

### 7.8.5.3 `lstimer_list_t lstimer_list[LSTIMER_LIST_LENGTH]` [static]

Our timer list.

Definition at line 36 of file `lstimer.c`.

### 7.8.5.4 `pthread_mutex_t lstimer_mutex` [static]

protect the timer list

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

## 7.8.5.5 pthread\_t ltimer\_thread [static]

the timer thread

Definition at line 38 of file ltimer.c.

## 7.8.5.6 timer\_t ltimer\_timerid [static]

our real time timer

Definition at line 41 of file ltimer.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 ltimer.c.

## 7.9 md2cmds.c File Reference

Implements commands to run the md2 diffractometer attached to a PMAC controlled by postgresql.

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

### Functions

- void [md2cmds\\_move\\_prep](#) (int mmask)  
*set up moving motors in a coordinate system*
- void [md2cmds\\_move\\_wait](#) (int mmask)  
*Wait for the movement to stop.*
- 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)
- void [md2cmds\\_transfer](#) ()  
*Transfer a sample.*
- void [md2cmds\\_moveAbs](#) (const char \*ccmd)  
*Move a motor to the position requested.*
- void [md2cmds\\_phase\\_change](#) (const char \*ccmd)  
*Move md2 devices to a preconfigured state.*
- 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\\_organs\\_prep](#) ()
- void [md2cmds\\_kappaphi\\_move](#) (double kappa\_deg, double phi\_deg)
- void [md2cmds\\_organs\\_wait](#) ()
- void [md2cmds\\_collect](#) ()  
*Collect some data.*
- void [md2cmds\\_rotate](#) ()  
*Spin 360 and make a video (recenter first, maybe)*
- void [md2cmds\\_rotate\\_cb](#) (char \*event)  
*Tell the database about the time we went through omega=zero.*
- void [md2cmds\\_maybe\\_rotate\\_done\\_cb](#) (char \*event)

- Now that we are done with the 360 rotation lets rehome right quick.*

  - void `md2cmds_set_scale_cb` (char \*event)

*Fix up xscale and yscale when zoom changes.*
- void `md2cmds_center` ()

*Move centering and alignment tables as requested TODO: Implement.*
- void `md2cmds_move_prep_done_cb` (char \*event)
  - void `md2cmds_time_capz_cb` (char \*event)

*Time the capillary motion for the transfer routine.*
- int `md2cmds_action_queue` (double timeout, char \*action)
  - void `md2cmds_action_wait` ()

*pause until md2cmds\_worker has finished running the command*
- void \* `md2cmds_worker` (void \*dummy)

*Our worker thread.*
- 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*
- pmac\_cmd\_queue\_t \* `md2cmds_moving_pq`  
*pmac queue item from last command*
- 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

### 7.9.1 Detailed Description

Implements commands to run the md2 diffractometer attached to a PMAC controlled by postgresql.

#### Date

2012

#### Author

Keith Brister

## Copyright

All Rights Reserved

Definition in file [md2cmds.c](#).

## 7.9.2 Function Documentation

7.9.2.1 `int md2cmds_action_queue ( double timeout, char * action )`

Definition at line 1083 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;
}

```

7.9.2.2 `void md2cmds_action_wait ( )`

pause until `md2cmds_worker` has finished running the command

Definition at line 1120 of file `md2cmds.c`.

```

{
    pthread_mutex_lock( &md2cmds_mutex);
    pthread_mutex_unlock( &md2cmds_mutex);
}

```

7.9.2.3 `void md2cmds_center ( )`

Move centering and alignment tables as requested TODO: Implement.

Definition at line 1041 of file `md2cmds.c`.

```

{
}

```

#### 7.9.2.4 void md2cmds\_collect ( )

Collect some data.

- < index of shot to be taken
- < start cnts
- < delta cnts
- < omega velocity cnts/msec
- < acceleration time (msec)
- < exposure time (msec)
- < one of the stages, at least, needs to be moved
- < unit to counts conversion
- < 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
- < combined motion mask to set up waiting

Definition at line 704 of file md2cmds.c.

```

{
    long long skey;
    double p170;
    double p171;
    double p173;
    double p175;
    double p180;
    int center_request;
    double u2c;
    double max_accel;
    double kappa_pos;
    double phi_pos;
    int motion_mask;

    u2c      = lsredis_getd( omega->u2c);
    max_accel = lsredis_getd( omega->max_accel);

    //
    // Put the organs into position
    //
    motion_mask = 16;

    md2cmds_move_prep( motion_mask);
    md2cmds_organs_move_presets( "In", "In", "In", "In",
        "Cover");
    md2cmds_move_wait( motion_mask);

    //
    // reset shutter has opened flag
    //
    lspmac_SockSendDPLine( NULL, "P3001=0 P3002=0");

    while( 1) {
        lspg_nextshot_call();

        motion_mask = 0;

        lspg_nextshot_wait();

        if( lspg_nextshot.no_rows_returned) {
            lspg_nextshot_done();
            break;
        }

        skey = lspg_nextshot.skey;
        lspg_query_push( NULL, "SELECT px.shots_set_state(%lld,
            'Preparing')", skey);

        center_request = 0;
        if( lspg_nextshot.active) {
            if(
                //
                // Don't move if we are within 0.1 microns of our destination

```



```

        //
        (fabs( lspg_nextshot.cx - cenx->position) >
0.1) ||
        (fabs( lspg_nextshot.cy - ceny->position) >
0.1) ||
        (fabs( lspg_nextshot.ax - alignx->position
) > 0.1) ||
        (fabs( lspg_nextshot.ay - aligny->position
) > 0.1) ||
        (fabs( lspg_nextshot.az - alignz->position
) > 0.1)) {

        motion_mask |= 6;
        center_request = 1;
        md2cmds_move_prep( 6);
        md2cmds_mvcenter_move( lspg_nextshot.
cx, lspg_nextshot.cy, lspg_nextshot.ax,
lspg_nextshot.ay, lspg_nextshot.az);
    }
}

// Maybe move kappa and/or phi
//
if( !lspg_nextshot.dsphi_isnull || !lspg_nextshot
.dskappa_isnull) {

    kappa_pos = lspg_nextshot.dskappa_isnull ?
lspmac_getPosition( kappa) : lspg_nextshot.
dskappa;
    phi_pos = lspg_nextshot.dsphi_isnull ?
lspmac_getPosition( phi) : lspg_nextshot.
dsphi;

    motion_mask |= 64;
    md2cmds_move_prep( 64);
    md2cmds_kappaphi_move( kappa_pos, phi_pos);
}

if( motion_mask)
    md2cmds_move_wait( motion_mask);

//
// Calculate the parameters we'll need to run the scan
//
p180 = lspg_nextshot.dsexp * 1000.0;
p170 = u2c * lspg_nextshot.sstart;
p171 = u2c * lspg_nextshot.dsowidth;
p173 = fabs(p180) < 1.e-4 ? 0.0 : u2c * lspg_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 has opened flag is down
// wait for the p3001=0 command to be noticed
//
pthread_mutex_lock( &lspmac_shutter_mutex);
if( lspmac_shutter_has_opened == 1)
    pthread_cond_wait( &lspmac_shutter_cond, &
lspmac_shutter_mutex);
pthread_mutex_unlock( &lspmac_shutter_mutex);

//
// Start the exposure
//

```

```

lspmac_SockSendDPlane( NULL, "&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);

//
// wait for the shutter to open
//
pthread_mutex_lock( &lspmac_shutter_mutex);
if( lspmac_shutter_has_opened == 0)
    pthread_cond_wait( &lspmac_shutter_cond, &
        lspmac_shutter_mutex);

//
// wait for the shutter to close
//
if( lspmac_shutter_state == 1)
    pthread_cond_wait( &lspmac_shutter_cond, &
        lspmac_shutter_mutex);
pthread_mutex_unlock( &lspmac_shutter_mutex);

//
// Signal the detector to start reading out
//
lspg_query_push( NULL, "SELECT px.unlock_diffractionmeter()");

//
// Update the shot status
//
lspg_query_push( NULL, "SELECT px.shots_set_state(%lld,
    'Writing')", skey);

//
// reset shutter has opened flag
//
lspmac_SockSendDPlane( NULL, "P3001=0");

//
// 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)) {

        center_request = 1;
        md2cmds_move_prep( 6);
        md2cmds_mvcenter_move( lspg_nextshot.
            cx, lspg_nextshot.cy, lspg_nextshot.ax,
            lspg_nextshot.ay, lspg_nextshot.az);
        md2cmds_move_wait(6);
    }
}
}
}

```

### 7.9.2.5 void md2cmds.init ( )

Initialize the md2cmds module.

Definition at line 1161 of file md2cmds.c.

```
{
```

```

memset( md2cmds_cmd, 0, sizeof( md2cmds_cmd));

pthread_mutex_init( &md2cmds_mutex, NULL);
pthread_cond_init( &md2cmds_cond, NULL);

pthread_mutex_init( &md2cmds_moving_mutex, NULL);
pthread_cond_init( &md2cmds_moving_cond, NULL);

md2cmds_md_status_code = lsredis_get_obj
( "md2_status_code");
lsredis_setstr( md2cmds_md_status_code, "
7");
}

```

#### 7.9.2.6 void md2cmds.kappaphi.move ( double *kappa\_deg*, double *phi\_deg* )

Definition at line 675 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);

}

```

#### 7.9.2.7 void md2cmds.maybe\_done\_moving.cb ( char \* *event* )

Track how many motors are moving.

Definition at line 639 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");

if( md2cmds_moving_count == 0)
pthread_cond_signal( &md2cmds_moving_cond);
pthread_mutex_unlock( &md2cmds_moving_mutex);

}

```

#### 7.9.2.8 void md2cmds.maybe\_rotate\_done.cb ( char \* *event* )

Now that we are done with the 360 rotation lets rehome right quick.

Definition at line 1005 of file md2cmds.c.

```

{
    if( rotating) {
        rotating = 0;
        lspmac_hommel_queue( omega);
    }
}

```

### 7.9.2.9 void md2cmds.move\_prep( int mmask )

set up moving motors in a coordinate system

Definition at line 33 of file md2cmds.c.

```

{
    int flag;

    pthread_mutex_lock( &lspmac_moving_mutex);
    flag = (lspmac_moving_flags & mmask) != 0;
    pthread_mutex_unlock( &lspmac_moving_mutex);

    //
    // Only wait for the all clear if it's not all clear already
    //
    if( flag) {
        //
        // Clear the motion flags for the given coordinate system(s)
        // Then set them.
        // Each time we wait until we've read back
        // the changed values
        //
        // This guarantees that when we are waiting for motion to stop that it did,
        // in fact, start
        //

        pthread_mutex_lock( &md2cmds_moving_mutex);
        md2cmds_moving_queue_wait = 1;
        pthread_mutex_unlock( &md2cmds_moving_mutex);

        //
        // Clear the centering and alignment stage flags
        //
        lspmac_SockSendDPLine( "move_prep", "M5075=(M5075 |
            %d) ^ %d", mmask, mmask);

        pthread_mutex_lock( &md2cmds_moving_mutex);
        while( md2cmds_moving_queue_wait)
            pthread_cond_wait( &md2cmds_moving_cond, &
                md2cmds_moving_mutex);
        pthread_mutex_unlock( &md2cmds_moving_mutex);

        //
        // Make sure the command propagates back to the status
        //
        pthread_mutex_lock( &lspmac_moving_mutex);
        while( (lspmac_moving_flags & mmask) != 0)
            pthread_cond_wait( &lspmac_moving_cond, &
                lspmac_moving_mutex);

        lslogging_log_message( "md2cmds_move_prep:
            lspmac_moving_flags = %d", lspmac_moving_flags);
        pthread_mutex_unlock( &lspmac_moving_mutex);
    }

    //
    // set a flag so the event listener doesn't look at zero motion before we
    // start and think we are done
    //
    pthread_mutex_lock( &md2cmds_moving_mutex);
    if( md2cmds_moving_count == 0)
        md2cmds_moving_count = -1;
    md2cmds_moving_queue_wait = 1;
    pthread_mutex_unlock( &md2cmds_moving_mutex);

    //
    // Now set the given motion flags
    //
    lspmac_SockSendDPLine( "move_prep", "M5075=(M5075 | %d) "
        , mmask);
}

```

```

pthread_mutex_lock( &pmac_queue_mutex);
//
// wait for the command to be sent
//
pthread_mutex_lock( &md2cmds_moving_mutex);
while( md2cmds_moving_queue_wait)
    pthread_cond_wait( &md2cmds_moving_cond, &
        md2cmds_moving_mutex);
pthread_mutex_unlock( &md2cmds_moving_mutex);

//
// Make sure it propagates
//
pthread_mutex_lock( &lspmac_moving_mutex);
while( (lspmac_moving_flags & mmask) != mmask)
    pthread_cond_wait( &lspmac_moving_cond, &
        lspmac_moving_mutex);

lslogging_log_message( "md2cmds_move_prep:
    lspmac_moving_flags = %d", lspmac_moving_flags);
pthread_mutex_unlock( &lspmac_moving_mutex);
}

```

#### 7.9.2.10 void md2cmds\_move\_prep\_done\_cb ( char \* event )

Definition at line 1044 of file md2cmds.c.

```

{
pthread_mutex_lock( &md2cmds_moving_mutex);
md2cmds_moving_queue_wait = 0;
pthread_cond_signal( &md2cmds_moving_cond);
pthread_mutex_unlock( &md2cmds_moving_mutex);
}

```

#### 7.9.2.11 void md2cmds\_move\_wait ( int mmask )

Wait for the movement to stop.

Definition at line 115 of file md2cmds.c.

```

{
//
// Just wait until the motion flags are lowered
// Note this does not mean the motors are done moving,
// just that the motion program is done.
//
// Look for the "In Position" events to see if we are really done
//
// We are assuming that the "Moving" callbacks were received
// before the motion programs have all finished. Probably a reasonable
// expectation but not really guaranteed
//

pthread_mutex_lock( &pmac_queue_mutex);
//
// wait for the command to be sent
//
if( md2cmds_moving_pq != NULL) {
    while( md2cmds_moving_pq->time_sent.tv_sec==0)
        pthread_cond_wait( &pmac_queue_cond, &pmac_queue_mutex
        );
}
pthread_mutex_unlock( &pmac_queue_mutex);

//
// Wait for the motion programs to finish
//
pthread_mutex_lock( &lspmac_moving_mutex);
while( lspmac_moving_flags & mmask)
    pthread_cond_wait( &lspmac_moving_cond, &
        lspmac_moving_mutex);
pthread_mutex_unlock( &lspmac_moving_mutex);

//
// Wait for the In Position events

```

```
//
pthread_mutex_lock( &md2cmds_moving_mutex);
while( md2cmds_moving_count > 0)
    pthread_cond_wait( &md2cmds_moving_cond, &
        md2cmds_moving_mutex);
pthread_mutex_unlock( &md2cmds_moving_mutex);
}
```

### 7.9.2.12 void md2cmds\_moveAbs ( const char \* ccmd )

Move a motor to the position requested.

#### Parameters

in	<i>ccmd</i>	The full command string to parse, ie, "moveAbs omega 180"
----	-------------	---

Definition at line 409 of file md2cmds.c.

```
{
char *cmd;
char *ignore;
char *ptr;
char *mtr;
char *pos;
double fpos;
char *endptr;
lspmac_motor_t *mp;
int i;

// ignore nothing
if( ccmd == NULL || *ccmd == 0) {
    return;
}

// 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;
}

// 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 error: missing
        motor name");
    free( cmd);
    return;
}

mp = NULL;
for( i=0; i<lspmac_nmotors; i++) {
    if( strcmp( lspmac_motors[i].name, mtr) == 0) {
        mp = &(lspmac_motors[i]);
        break;
    }
}
if( mp == NULL) {
    lslogging_log_message( "md2cmds moveAbs error: cannot
        find motor %s", mtr);
    free( cmd);
    return;
}

pos = strtok_r( NULL, " ", &ptr);
if( pos == NULL) {
    lslogging_log_message( "md2cmds moveAbs error: missing
        position");
    free( cmd);
    return;
}
```

```

}

fpos = strtod( pos, &endptr);
if( pos == endptr) {
    //
    // Maybe we have a preset. Give it a whirl
    // In any case we are done here.
    //
    lspmac_move_preset_queue( mp, pos);
    free( cmd);
    return;
}

if( mp != NULL && mp->moveAbs != NULL) {
    wprintw( term_output, "Moving %s to %f\n", mtr, fpos);
    wnoutrefresh( term_output);
    mp->moveAbs( mp, fpos);
}

free( cmd);
}

```

#### 7.9.2.13 void md2cmds\_mvcenter\_move ( double cx, double cy, double ax, double ay, double az )

Move the centering and alignment tables.

##### Parameters

in	cx	Requested Centering Table X
in	cy	Requested Centering Table Y
in	ax	Requested Alignment Table X
in	ay	Requested Alignment Table Y
in	az	Requested Alignment Table Z

Definition at line 611 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( cenx, 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);
}

```

#### 7.9.2.14 void md2cmds\_organs\_move\_presets ( char \* pay, char \* paz, char \* pcy, char \* pcz, char \* psz )

Definition at line 173 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);

//
// 170          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);
}

```

### 7.9.2.15 void md2cmds\_organs\_prep ( )

Definition at line 667 of file md2cmds.c.

```

{
//
// we are coordinate system 5, mask is 1 << (cs - 1)
//
md2cmds_move_prep( 16);
}

```

### 7.9.2.16 void md2cmds\_organs\_wait ( )

Definition at line 694 of file md2cmds.c.

```

{
//
// we are coordinate system 5, mask is 1 << (cs - 1)
//
md2cmds_move_wait( 16);
}

```



## 7.9.2.17 void 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 496 of file md2cmds.c.

```

{
char *cmd;
char *ignore;
char *ptr;
char *mode;

if( ccmd == NULL || *ccmd == 0)
    return;

// 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;
}

//
// 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");
    free( cmd);
    return;
}

if( strcmp( mode, "manualMount") == 0) {
    lspmac_move_or_jog_preset_queue( kappa,
        "manualMount", 1);
    lspmac_move_or_jog_preset_queue( omega,
        "manualMount", 0);
    lspmac_move_or_jog_abs_queue( phi, 0.0, 0)
        ;
    lspmac_move_or_jog_preset_queue( aperz,
        "Cover", 1);
    lspmac_move_or_jog_preset_queue( capz,
        "Cover", 1);
    lspmac_move_or_jog_preset_queue( scint,
        "Cover", 1);
    md2cmds_moveAbs( "moveAbs backLight 0");
    md2cmds_moveAbs( "moveAbs backLight.intensity 0");
    md2cmds_moveAbs( "moveAbs cryo 1");
    md2cmds_moveAbs( "moveAbs fluo 0");
    md2cmds_moveAbs( "moveAbs cam.zoom 1");
} else if( strcmp( mode, "robotMount") == 0) {
    lspmac_hommel_queue( kappa);
    lspmac_hommel_queue( omega);
    lspmac_move_or_jog_abs_queue( phi, 0.0, 0);
    lspmac_move_or_jog_preset_queue( apery,
        "In", 1);
    lspmac_move_or_jog_preset_queue( aperz,
        "In", 1);
    lspmac_move_or_jog_preset_queue( capz,
        "Cover", 1);
    lspmac_move_or_jog_preset_queue( scint,
        "Cover", 1);
    md2cmds_moveAbs( "moveAbs backLight 0");
    md2cmds_moveAbs( "moveAbs backLight.intensity 0");
    md2cmds_moveAbs( "moveAbs cryo 1");
    md2cmds_moveAbs( "moveAbs fluo 0");
    md2cmds_moveAbs( "moveAbs cam.zoom 1");
} else if( strcmp( mode, "center") == 0) {
    md2cmds_moveAbs( "moveAbs kappa 0");
    md2cmds_moveAbs( "moveAbs omega 0");
    lspmac_move_or_jog_abs_queue( phi, 0.0,

```

```

    0);
    lspmac_move_or_jog_preset_queue( apery,
        "In", 1);
    lspmac_move_or_jog_preset_queue( aperz,
        "In", 1);
    lspmac_move_or_jog_preset_queue( capy,
        "In", 1);
    lspmac_move_or_jog_preset_queue( capz,
        "In", 1);
    lspmac_move_or_jog_preset_queue( scint,
        "Cover", 1);
    md2cmds_moveAbs( "moveAbs backLight 1");
    md2cmds_moveAbs( "moveAbs cam.zoom 1");
    md2cmds_moveAbs( "moveAbs cryo 0");
    md2cmds_moveAbs( "moveAbs fluo 0");
} else if( strcmp( mode, "dataCollection") == 0) {
    lspmac_move_or_jog_preset_queue( apery,
        "In", 1);
    lspmac_move_or_jog_preset_queue( aperz,
        "In", 1);
    lspmac_move_or_jog_preset_queue( capy,
        "In", 1);
    lspmac_move_or_jog_preset_queue( capz,
        "In", 1);
    lspmac_move_or_jog_preset_queue( scint,
        "Cover", 1);
    md2cmds_moveAbs( "moveAbs backLight 0");
    md2cmds_moveAbs( "moveAbs backLight.intensity 0");
    md2cmds_moveAbs( "moveAbs cryo 0");
    md2cmds_moveAbs( "moveAbs fluo 0");
} else if( strcmp( mode, "beamLocation") == 0) {
    md2cmds_moveAbs( "moveAbs kappa 0");
    md2cmds_moveAbs( "moveAbs omega 0");
    lspmac_move_or_jog_preset_queue( apery,
        "In", 1);
    lspmac_move_or_jog_preset_queue( aperz,
        "In", 1);
    lspmac_move_or_jog_preset_queue( capy,
        "In", 1);
    lspmac_move_or_jog_preset_queue( capz,
        "In", 1);
    lspmac_move_or_jog_preset_queue( scint,
        "Scintillator", 1);
    md2cmds_moveAbs( "moveAbs backLight 0");
    md2cmds_moveAbs( "moveAbs cam.zoom 1");
    md2cmds_moveAbs( "moveAbs cryo 0");
    md2cmds_moveAbs( "moveAbs fluo 0");
} else if( strcmp( mode, "safe") == 0) {
    md2cmds_moveAbs( "moveAbs kappa 0");
    md2cmds_moveAbs( "moveAbs omega 0");
    lspmac_move_or_jog_preset_queue( apery,
        "In", 1);
    lspmac_move_or_jog_preset_queue( aperz,
        "Cover", 1);
    lspmac_move_or_jog_preset_queue( capy,
        "In", 1);
    lspmac_move_or_jog_preset_queue( capz,
        "Cover", 1);
    lspmac_move_or_jog_preset_queue( scint,
        "Cover", 1);
    md2cmds_moveAbs( "moveAbs backLight 0");
    md2cmds_moveAbs( "moveAbs cam.zoom 1");
    md2cmds_moveAbs( "moveAbs cryo 0");
    md2cmds_moveAbs( "moveAbs fluo 0");
}

free( cmd);
}

```

### 7.9.2.18 double md2cmds\_prep\_axis ( lspmac\_motor\_t \* mp, double pos )

Definition at line 156 of file md2cmds.c.

```

{
    double rtn;
    double u2c;

    pthread_mutex_lock( &(mp->mutex));
    u2c = lsredis_getd( mp->u2c);

    rtn = u2c * pos;
}

```

```

mp->motion_seen = 0;
mp->not_done    = 1;
pthread_mutex_unlock( &(mp->mutex));

return rtn;
}

```

### 7.9.2.19 void md2cmds\_rotate ( )

Spin 360 and make a video (recenter first, maybe)

Definition at line 892 of file md2cmds.c.

```

double cx, cy, ax, ay, az;

//
// BLUMax disables scintillator here.
//

//
// get the new center information
//
lslogging_log_message( "md2cmds_rotate: calling
    getcenter");
lspg_getcenter_call();

lslogging_log_message( "md2cmds_rotate: wait for
    getcenter");
lspg_getcenter_wait();

lslogging_log_message( "md2cmds_rotate: moving backlight
    up");
// put up the back light
blight_ud->moveAbs( blight_ud, 1);

if( lspg_getcenter.no_rows_returned) {
    //
    // Always specify zoom even if no other center information is found
    //
    zoom->moveAbs( zoom, 1);    // default zoom is 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) {
        zoom->moveAbs( zoom, lspg_getcenter.zoom
        );
    } else {
        zoom->moveAbs( zoom, 1);
    }

    //
    // Grab the current positions and perhaps add the tad specified by
    // getcenter
    //
    cx = lspmac_getPosition( cenx);
    cy = lspmac_getPosition( ceny);
    ax = lspmac_getPosition( alignx);
    ay = lspmac_getPosition( aligny);
    az = lspmac_getPosition( 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.dcx_isnull == 0)
        cx += lspg_getcenter.dcx;

    if( lspg_getcenter.dcy_isnull == 0)
        cy += lspg_getcenter.dcy;

    if( lspg_getcenter.dax_isnull == 0)
        ax += lspg_getcenter.dax;

    if( lspg_getcenter.day_isnull == 0)
        ay += lspg_getcenter.day;

    if( lspg_getcenter.daz_isnull == 0)
        az += lspg_getcenter.daz;
}

```

```

lslogging_log_message( "md2cmds_rotate: requested
    positions cx %f, cy %f, ax %f, ay %f, az %f", cx, cy, ax, ay, az);

md2cmds_move_prep( 6);
lslogging_log_message( "md2cmds_rotate: moving center"
);
md2cmds_mvcenter_move( cx, cy, ax, ay, az);

lslogging_log_message( "md2cmds_rotate: waiting for
    center move");
md2cmds_move_wait(6);
lslogging_log_message( "md2cmds_rotate: done waiting")
;
}
lspg_getcenter_done();

// Omega was just homed before we mounted the sample, don't do it again here

// Report new center positions
cx = lspmac_getPosition( cenx);
cy = lspmac_getPosition( ceny);
ax = lspmac_getPosition( alignx);
ay = lspmac_getPosition( aligny);
az = lspmac_getPosition( alignz);
lspg_query_push( NULL, "SELECT px.applycenter( %.3f, %.3f,
    %.3f, %.3f, %.3f, %.3f, %.3f)", cx, cy, ax, ay, az, lspmac_getPosition
    (kappa), lspmac_getPosition( phi));

lspmac_moveabs_wait( zoom);

lslogging_log_message( "md2cmds_rotate: done with
    applycenter");
lspmac_video_rotate( 4.0);
lslogging_log_message( "md2cmds_rotate: starting
    rotation");
rotating = 1;
}

```

#### 7.9.2.20 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 988 of file md2cmds.c.

```

{
struct tm t;
int usecs;

localtime_r( &(omega_zero_time.tv_sec), &t);

lslogging_log_message( "md2cmds_rotate_cb: Here I am");

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)));
}

```

#### 7.9.2.21 void md2cmds\_run( )

Start up the thread.

Definition at line 1176 of file md2cmds.c.

```

{
pthread_create( &md2cmds_thread, NULL,
    md2cmds_worker, NULL);
lsevents_add_listener( "omega crossed zero",

```

```

    md2cmds_rotate_cb);
    lsevents_add_listener( ".+ (Moving|In Position)",
        md2cmds_maybe_done_moving_cb);
    lsevents_add_listener( "capz (Moving|In Position)",
        md2cmds_time_capz_cb);
    lsevents_add_listener( "move_prep command done",
        md2cmds_move_prep_done_cb);
}

```

#### 7.9.2.22 void md2cmds\_set\_scale\_cb ( char \* event )

Fix up xscale and yscale when zoom changes.

Definition at line 1015 of file md2cmds.c.

```

{
    int mag;
    lsredis_obj_t *p1, *p2;
    char *vp;

    mag = lspmac_getPosition( zoom);

    p1 = lsredis_get_obj( "cam.xScale");
    p2 = lsredis_get_obj( "cam.zoom.%d.ScaleX", mag);

    vp = lsredis_getstr( p2);
    lsredis_setstr( p2, vp);
    free( vp);

    p1 = lsredis_get_obj( "cam.yScale");
    p2 = lsredis_get_obj( "cam.zoom.%d.ScaleY", mag);

    vp = lsredis_getstr( p2);
    lsredis_setstr( p2, vp);
    free( vp);
}

```

#### 7.9.2.23 void md2cmds\_time\_capz\_cb ( char \* event )

Time the capillary motion for the transfer routine.

< track the time spent moving capz

Definition at line 1053 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 / 1000000000.0;
    }
}

```

#### 7.9.2.24 void md2cmds\_transfer ( )

Transfer a sample.

Definition at line 231 of file md2cmds.c.

```

    {
int nextsample, abort_now;
double esttime;
double ax, ay, az, cx, cy, horz, vert, oref;
int err;
int motion_mask;

nextsample = lspg_nextsample_all( &err);
if( err) {
    lslogging_log_message( "md2cmds_transfer: no sample
        requested to be transfered, false alarm");
    return;
}

//
// BLUMax sets up an abort dialogbox here. Probably we should figure out how
// we are going to handle that.
//

// Wait for everything to stop moving
// TODO: timeout and abort if we are moving forever
//
md2cmds_move_wait( 0);

pthread_mutex_lock( &md2cmds_moving_mutex);
while( md2cmds_moving_count > 0)
    pthread_cond_wait( &md2cmds_moving_cond, &
        md2cmds_moving_mutex);
pthread_mutex_unlock( &md2cmds_moving_mutex);

//
// 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);

if( lsredis_getd( capz->u2c) <= 0.0 || lsredis_getd
    ( capz->max_speed) <= 0.0 || lsredis_getd( capz->
    max_accel) <= 0.0) {
    esttime = 0.0;
} else {

    // Here we assume moving the capillary is the rate limiting step in
    // preparing the MD2.
    //
    // TODO: look at factors in which something besides the capillary determines
    // the time.
    //
    // pretend we are going to zero instead of the "Out" position. This should
    // be less than a 5% error
    // and is probably not too horrible
    //
    // This also treats S curve acceleration as taking the same time as linear
    // acceleration.
    //
    esttime = lspmac_getPosition( capz)/lsredis_getd
        ( capz->u2c)/(lsredis_getd( capz->max_speed));
    // Time if we moved at constant velocity
    esttime += lsredis_getd( capz->max_speed)/
        lsredis_getd(capz->max_accel);
    // Correction for time spent accelerating
    esttime /= 1000.;
    // convert from milliseconds to seconds
}

lspg_starttransfer_call( nextsample,
    lspmac_getBIPosition( sample_detected), ax,
    ay, az, horz, vert, esttime);

// put the light down if it's not already
//
if( lspmac_getBIPosition( blight_down) != 1)
    blight_ud->moveAbs( blight_ud, 0);

// Pull the fluorescence detector out of the way
//
if( lspmac_getBIPosition( fluo_back) != 1)
    blight_ud->moveAbs( fluo, 0);

```

```

// get ready to move the organs, omega, kappa, and phi
//      omega      organs      kappa/phi
motion_mask = 1 | 16 | 64;
md2cmds_move_prep( motion_mask);
//
// Put the organs into position
//
md2cmds_organs_move_presets( "In", "Cover", "In",
    "Cover", "Cover");

//
// Home Kappa
//
lspmac_home1_queue( kappa);

//
// Home omega
//
lspmac_home1_queue( omega);

//
// wait for kappa cause we can't home phi until kappa's done
//
lspmac_moveabs_wait( kappa);

//
// Home phi (whatever that means)
//
lspmac_home1_queue( phi);
{
    pmac_cmd_queue_t *mypq;
    //
    // Do a little dance to have the md2cmds_moving routines see the
    // last move command we sent to the pmac
    //
    // try not to grab too many mutexs at the same time to lower the chance of
    // a deadlock.
    //
    pthread_mutex_lock( &phi->mutex);
    mypq = phi->pq;
    pthread_mutex_unlock( &phi->mutex);

    pthread_mutex_lock( &md2cmds_moving_mutex);
    md2cmds_moving_pq = mypq;
    pthread_mutex_unlock( &md2cmds_moving_mutex);
}

// 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
//
if( lspg_starttransfer.no_rows_returned ||
    lspg_starttransfer.starttransfer != 1)
    abort_now = 1;
else
    abort_now = 0;

lspg_starttransfer_done();

//
// Wait for all those motors to stop moving
//
md2cmds_move_wait( motion_mask);

// TODO: check that all the motors are where we told them to go
//

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");
    return;
}

// refuse to go on if we do not have positive confirmation that the backlight
// is down and the
// fluorescence detector is back
//
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 fluoescence dectector is not back");
    return;
}

```

```

//
// 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);

// simplest query yet!
lspg_query_push( lspg_waitcryo_cb, "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();
}

```

#### 7.9.2.25 void\* md2cmds\_worker( void \* dummy )

Our worker thread.

##### Parameters

<i>dummy</i>	[in] Unused but required by protocol
--------------	--------------------------------------

Definition at line 1127 of file md2cmds.c.

```

{
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
        );

    if( strcmp( md2cmds_cmd, "transfer") == 0) {
        md2cmds_transfer();
    } else if( strcmp( md2cmds_cmd, "collect") == 0) {
        md2cmds_collect();
    } else if( strcmp( md2cmds_cmd, "rotate") == 0) {
        md2cmds_rotate();
    } else if( strcmp( md2cmds_cmd, "center") == 0) {
        md2cmds_center();
    } else if( strncmp( md2cmds_cmd, "moveAbs", 7) == 0) {
        md2cmds_moveAbs( md2cmds_cmd);
    } else if( strncmp( md2cmds_cmd, "changeMode", 10) == 0) {
        md2cmds_phase_change( md2cmds_cmd);
    }

    md2cmds_cmd[0] = 0;
}
}

```

### 7.9.3 Variable Documentation

#### 7.9.3.1 double md2cmds\_capz\_moving\_time = NAN [static]

Definition at line 28 of file md2cmds.c.



#### 7.9.3.2 `char md2cmds_cmd[MD2CMDS_CMD_LENGTH]`

our command;

Definition at line 20 of file md2cmds.c.

#### 7.9.3.3 `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.

#### 7.9.3.4 `lsredis_obj_t* md2cmds_md_status_code`

Definition at line 22 of file md2cmds.c.

#### 7.9.3.5 `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.

#### 7.9.3.6 `int md2cmds_moving_count = 0`

Definition at line 18 of file md2cmds.c.

#### 7.9.3.7 `pthread_mutex_t md2cmds_moving_mutex`

message passing between md2cmds and pg

Definition at line 15 of file md2cmds.c.

#### 7.9.3.8 `pmac_cmd_queue_t* md2cmds_moving_pq`

pmac queue item from last command

Definition at line 16 of file md2cmds.c.

#### 7.9.3.9 `int md2cmds_moving_queue_wait = 0`

Definition at line 13 of file md2cmds.c.

#### 7.9.3.10 `pthread_mutex_t md2cmds_mutex`

mutex for the condition

Definition at line 11 of file md2cmds.c.

#### 7.9.3.11 `pthread_t md2cmds_thread` `[static]`

Definition at line 24 of file md2cmds.c.

### 7.9.3.12 `int rotating = 0` `[static]`

flag: when omega is in position after a rotate we want to re-home omega

Definition at line 26 of file md2cmds.c.

## 7.10 `mk_pgpmac_redis.py` File Reference

### Namespaces

- namespace `mk_pgpmac_redis`

### 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.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"`
- 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")`

## 7.11 `pgpmac.c` File Reference

Main for the pgpmac project.

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

### Functions

- void `stdinService` (struct pollfd \*evt)  
*Handle keyboard input.*
- void `pgpmac_printf` (char \*fmt,...)  
*Terminal output routine ala printf.*
- 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*
- 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 Function Documentation

#### 7.11.2.1 int main ( int *argc*, char \*\* *argv* )

Our main routine.

#### Parameters

in	<i>argc</i>	Number of arguments
in	<i>argv</i>	Vector of argument strings

Definition at line 351 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

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;

    }
}

stdinfd.fd = 0;
stdinfd.events = POLLIN;

initscr();                            // Start ncurses
raw();                                // Line buffering disabled, control
    chars trapped
keypad( stdscr, TRUE);                // Why is F1 nifty?
refresh();

pthread_mutex_init( &ncurses_mutex, NULL);    // don't lock
    this mutex yet because we are not multi-threaded until the "_run" functions

//
// Since the modules reference objects in other modules it is important
// that everyone is initialized before anyone runs
//
lslogging_init();
lslogging_run();
lsevents_init();
lsevents_run();
lstimer_init();
lstimer_run();
lsredis_init( "MD2-21-ID-E", "redis\\.kvseq|stns\\.2\\.\\. (\\+)", "
    stns.2");
lsredis_run();

lspmac_init( ivars, mvars);
lspg_init();
md2cmds_init();

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);
scrollok( 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();

lspmac_run();
lspg_run();
md2cmds_run();

while( running) {
    //

```

```

// Big loop
//

nfd = 0;

//
// keyboard
//
memcpy( &(fda[nfd++]), &stdinfd, sizeof( struct pollfd));

if( nfd == 0) {
    //
    // No connections 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;
}

```

### 7.11.2.2 void pgpmac\_printf ( char \*fmt, ... )

Terminal output routine ala printf.

#### Parameters

in	fmt	Printf style formatting string
----	-----	--------------------------------

Definition at line 328 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);
    wnoutrefresh( term_input);
    doupdate();

    pthread_mutex_unlock( &ncurses_mutex);
}

```

### 7.11.2.3 void stdinService ( struct pollfd \*evt )

Handle keyboard input.

#### Parameters

in	evt	The pollfd object that caused this call
----	-----	---

Definition at line 254 of file pgpmac.c.

```

    {
static char cmds[1024];
static char cntrlcmd[2];
static unsigned int cmds_on = 0;
int ch;

for( ch=wgetch(term_input); ch != ERR && running; ch=wgetch(
    term_input)) {
    // wprintw( term_output, "%04x\n", ch);
    // wnoutrefresh( term_output);

    switch( ch) {
case KEY_F(1):
case KEY_F(2):
case KEY_F(3):
    running = 0;
    break;

case 0x0001:      // Control-A
case 0x0002:      // Control-B
case 0x0003:      // Control-C
case 0x0004:      // Control-D
case 0x0005:      // Control-E
case 0x0006:      // Control-F
case 0x0007:      // Control-G
case 0x000b:      // Control-K
case 0x000f:      // Control-O
case 0x0010:      // Control-P
case 0x0011:      // Control-Q
case 0x0012:      // Control-R
case 0x0013:      // Control-Q
case 0x0016:      // Control-V
    cntrlcmd[0] = ch;
    cntrlcmd[1] = 0;
    lspmac_SockSendline( NULL, cntrlcmd);
    // PmacSockSendControlCharPrint( ch);
    break;

case KEY_BACKSPACE:
    cmds[cmds_on] = 0;
    cmds_on == 0 ? 0 : cmds_on--;
    break;

case KEY_ENTER:
case 0x000a:
    if( cmds_on > 0 && strlen( cmds) > 0) {
        lspmac_SockSendline( NULL, cmds);
    }
    memset( cmds, 0, sizeof(cmds));
    cmds_on = 0;
    break;

default:
    if( cmds_on < sizeof( cmds)-1) {
        cmds[cmds_on++] = ch;
        cmds[cmds_on] = 0;
    }
    break;
    }

if( running) {
    mvwprintw( term_input, 1, 1, "PMAC> %s", cmds);
    wclrtoeol( term_input);
    box( term_input, 0, 0);
    wnoutrefresh( term_input);
    doupdate();
}
}
}

```

## 7.11.3 Variable Documentation

### 7.11.3.1 pthread\_mutex\_t ncurses\_mutex

allow more than one thread access to the screen

Definition at line 242 of file pgpmac.c.

### 7.11.3.2 `int running = 1` `[static]`

Definition at line 249 of file pgpmac.c.

### 7.11.3.3 `struct pollfd stdinfd` `[static]`

Handle input from the keyboard.

Definition at line 248 of file pgpmac.c.

### 7.11.3.4 `WINDOW* term_input`

place to put the cursor

Definition at line 238 of file pgpmac.c.

### 7.11.3.5 `WINDOW* term_output`

place to print stuff out

Definition at line 237 of file pgpmac.c.

### 7.11.3.6 `WINDOW* term_status`

shutter, lamp, air, etc status

Definition at line 239 of file pgpmac.c.

### 7.11.3.7 `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>
```

## 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 it's 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 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 it's 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_nextsample_struct lspg_nextsample_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 lspmac_getPosition (lspmac_motor_t *)`  
*get the motor position (with locking)*
- `char ** lspg_array2ptrs (char *)`  
*returns a null terminated list of strings parsed from postgresql array*
- `char ** lsredis_get_string_array (lsredis_obj_t *p)`

- void [lspmac\\_SockSendDPLine](#) (char \*, char \*fmt,...)  
*prepare (queue up) a line to send the dpram ascii command interface*
- [pmac\\_cmd\\_queue\\_t \\* lspmac\\_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*
- void [PmacSockSendline](#) (char \*s)
- unsigned int [lspg\\_nextsample\\_all](#) (int \*err)
- 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 [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 [lspg\\_demandairrights\\_all](#) ()  
*do nothing until we get airrights*
- void [lspg\\_getcenter\\_call](#) ()  
*Request a getcenter query.*
- void [lspg\\_getcenter\\_done](#) ()  
*Done with getcenter query.*
- void [lspg\\_getcenter\\_wait](#) ()  
*Wait for a getcenter query to return.*
- void [lspg\\_getcurrentsampleid\\_wait\\_for\\_id](#) (unsigned int test)
- void [lspg\\_init](#) ()  
*Initiallize the lspg module.*
- void [lspg\\_nextshot\\_call](#) ()  
*Queue up a nextshot query.*
- void [lspg\\_nextshot\\_done](#) ()  
*Called when the next shot query has been processed.*
- void [lspg\\_nextshot\\_wait](#) ()  
*Wait for the next shot query to get processed.*
- 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)  
*Convinence 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](#) ()
- 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\\_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\\_init](#) (int, int)
  - Initialize this module.*
- void [lspmac\\_jogabs\\_queue](#) ([lspmac\\_motor\\_t](#) \*, double)
  - Use jog to move motor to requested position.*
- void [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.*
- void [lspmac\\_move\\_or\\_jog\\_preset\\_queue](#) ([lspmac\\_motor\\_t](#) \*, char \*, int)
  - move using a preset value*
- void [lspmac\\_move\\_or\\_jog\\_queue](#) ([lspmac\\_motor\\_t](#) \*, double, int)
- void [lspmac\\_move\\_preset\\_queue](#) ([lspmac\\_motor\\_t](#) \*mp, char \*preset\_name)
  - Move a given motor to one of its preset positions.*
- void [lspmac\\_moveabs\\_queue](#) ([lspmac\\_motor\\_t](#) \*, double)
  - Use coordinate system motion program, if available, to move motor to requested position.*
- void [lspmac\\_moveabs\\_wait](#) ([lspmac\\_motor\\_t](#) \*mp)
  - 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)
- int [lsredis\\_getb](#) ([lsredis\\_obj\\_t](#) \*p)
- double [lsredis\\_getd](#) ([lsredis\\_obj\\_t](#) \*p)
- void [lsredis\\_init](#) (char \*pub, char \*re, char \*head)
  - Initialize this module, that is, set up the connections.*
- int [lsredis\\_regexec](#) (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 [lstimer\\_add\\_timer](#) (char \*, int, unsigned long int, unsigned long int)
  - Create a timer.*
- void [lstimer\\_init](#) ()
  - Initialize the timer list and pthread stuff.*
- void [lstimer\\_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.*

## 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*
- `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 X.*
- `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 scintillator 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`

### 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 MD2CMDS_CMD_LENGTH 32`

Definition at line 478 of file pgpmac.h.

### 7.12.3 Typedef Documentation

7.12.3.1 `typedef struct lspg_demandairrights_struct lspg_demandairrights_t`

7.12.3.2 `typedef struct lspg_getcenter_struct lspg_getcenter_t`

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

7.12.3.3 `typedef struct lspg_getcurrentsampleid_struct lspg_getcurrentsampleid_t`

7.12.3.4 `typedef struct lspg_nextsample_struct lspg_nextsample_t`

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

7.12.3.5 `typedef struct lspg_nextshot_struct lspg_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 lspgQueryQueueStruct lspg_query_queue_t`

Store each query along with it's callback function.

All calls are asynchronous

7.12.3.7 `typedef struct lspg_starttransfer_struct lspg_starttransfer_t`

returns 1 if transfer can continue 0 to abort

7.12.3.8 `typedef struct lspg_waitcryo_struct lspg_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 sychronized with our redis db.

7.12.3.12 `typedef struct lspmac_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 void lsevents\_add\_listener ( char \* event, void(\*)(char \*) cb )

Add a callback routine to listen for a specific event.

#### Parameters

<i>event</i>	the name of the event to listen for
<i>cb</i>	the routine to call

Definition at line 75 of file lsevents.c.

```

{
    lsevents_listener_t *new;
    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, event, 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( event);
    new->cb = cb;

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

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

```

### 7.12.4.2 void lsevents\_init ( )

Initialize this module.

Definition at line 214 of file lsevents.c.

```

{
    pthread_mutex_init( &lsevents_queue_mutex, NULL);

```

```
pthread_cond_init( &lsevents_queue_cond, NULL);
pthread_mutex_init( &lsevents_listener_mutex, NULL);
}
```

#### 7.12.4.3 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
<i>cb</i>	The callback routine to remove

Definition at line 120 of file lsevents.c.

```

{

lsevents_listener_t *last, *current;

//
// 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;
    }
}
pthread_mutex_unlock( &lsevents_listener_mutex);

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

#### 7.12.4.4 void lsevents\_run ( )

Start up the thread and get out of the way.

Definition at line 222 of file lsevents.c.

```

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

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

Call the callback routines for the given event.

##### Parameters

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

Definition at line 45 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);

    lslogging_log_message( "lsevents_send_event: %s", event)
        ;

    // 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);
}

```

#### 7.12.4.6 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");
}

```

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

The routine everyone will be talking about.

##### Parameters

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

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);
}

```

#### 7.12.4.8 void lslogging\_run ( )

Start up the worker thread.

Definition at line 105 of file lslogging.c.

```

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

```

#### 7.12.4.9 char\*\* lspg\_array2ptrs ( char \* )

returns a null terminated list of strings parsed from postgresql array

Definition at line 161 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);
}

```

#### 7.12.4.10 void lspg\_demandairrights\_all( )

do nothing until we get airrights

Definition at line 556 of file lspg.c.

```

{
    lspg_demandairrights_call();
}

```

```

lspg_demandairrights_wait();
// there is no "done" version
}

```

#### 7.12.4.11 void lspg\_getcenter\_call ( )

Request a getcenter query.

Definition at line 1177 of file lspg.c.

```

{
pthread_mutex_lock( &lspg_getcenter.mutex);
lspg_getcenter.new_value_ready = 0;
pthread_mutex_unlock( &lspg_getcenter.mutex);

lspg_query_push( lspg_getcenter_cb, "SELECT *
FROM px.getcenter2() ");
}

```

#### 7.12.4.12 void lspg\_getcenter\_done ( )

Done with getcenter query.

Definition at line 1195 of file lspg.c.

```

{
pthread_mutex_unlock( &(lspg_getcenter.mutex));
}

```

#### 7.12.4.13 void lspg\_getcenter\_wait ( )

Wait for a getcenter query to return.

Definition at line 1187 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));
}

```

#### 7.12.4.14 void lspg\_getcurrentsampleid\_wait\_for\_id ( unsigned int test )

Definition at line 393 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
);
}

```

**7.12.4.15 void lspg\_init ( )**

Initialize the lspg module.

Definition at line 1758 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_diffractionmeter_init();
lspg_nextsample_init();
lspg_nextshot_init();
lspg_seq_run_prep_init();
lspg_starttransfer_init();
lspg_wait_for_detector_init();
lspg_waitcryo_init();
}

```

**7.12.4.16 unsigned int lspg\_nextsample\_all ( int \* err )**

Definition at line 468 of file lspg.c.

```

{
unsigned int rtn;

lspg_nextsample_call();
lspg_nextsample_wait();

if( lspg_nextsample.no_rows_returned) {
    rtn = 0;
    *err = 1;
} else {
    if( lspg_nextsample.nextsample_isnull) {
        rtn = 0;
        *err = 1;
    } else {
        rtn = lspg_nextsample.nextsample;
        *err = 0;
    }
}
lspg_nextsample_done();

return rtn;
}

```

**7.12.4.17 void lspg\_nextshot\_call ( )**

Queue up a nextshot query.

Definition at line 824 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()");
}

```

**7.12.4.18 void lspg\_nextshot\_done ( )**

Called when the next shot query has been processed.

Definition at line 842 of file lspg.c.

```

    {
        pthread_mutex_unlock( &(lspg_nextshot.mutex));
    }

```

#### 7.12.4.19 void lspg\_nextshot\_wait ( )

Wait for the next shot query to get processed.

Definition at line 834 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));
    }

```

#### 7.12.4.20 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 234 of file kvredis.c.

```

    {
        int idx;
        va_list arg_ptr;

        //
        // Pause the thread while we service the queue
        //
        if( (lspg_query_queue_on + 1) % LS_PG_QUERY_QUEUE_LENGTH
            == lspg_query_queue_off % LS_PG_QUERY_QUEUE_LENGTH
        ) {
            fprintf( stderr, "lspg_query_push: queue is full. Ignoring query \"%s\"\n"
                , fmt);
            return;
        }

        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++;
    }
};

```

#### 7.12.4.21 void lspg\_run ( )

Start 'er runnin'.

Definition at line 1777 of file lspg.c.

```

    {
        pthread_create( &lspg_thread, NULL, lspg_worker, NULL);
        lsevents_add_listener( "Sample(Detected|Absent)",

```



```
    lspmac_sample_detector_cb);
}
```

**7.12.4.22** void `lspg_seq_run_prep_all` ( long long *skey*, double *kappa*, double *phi*, double *cx*, double *cy*, double *ax*, double *ay*, double *az* )

Convenience 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 1095 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();
    }
```

**7.12.4.23** 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 302 of file `lspg.c`.

```

                                                                    {
pthread_mutex_lock( &(amp;lspg_starttransfer.mutex));
lspg_starttransfer.new_value_ready = 0;
pthread_mutex_unlock( &(amp;lspg_starttransfer.mutex));

lspg_query_push( lspg_starttransfer_cb, "
    SELECT px.starttransfer( %d, %d, %.3f, %.3f, %.3f, %.3f, %.3f, %.3f",
        nextsample, sample_detected, ax, ay, az, horz
        , vert, esttime);
}
```

**7.12.4.24** void `lspg_starttransfer.done` ( )

Definition at line 317 of file `lspg.c`.

```
    {
pthread_mutex_unlock( &(amp;lspg_starttransfer.mutex));
}
```

**7.12.4.25** void `lspg_starttransfer.wait` ( )

Definition at line 311 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));
}

```

#### 7.12.4.26 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

Definition at line 507 of file lspg.c.

```

    {
pthread_mutex_lock( &lspg_waitcryo.mutex);
lspg_waitcryo.new_value_ready = 0;

lspg_query_push( lspg_waitcryo_cb, "SELECT
px.waitcryo()");

while( lspg_waitcryo.new_value_ready == 0)
pthread_cond_wait( &lspg_waitcryo.cond, &lspg_waitcryo
.mutex);

pthread_mutex_unlock( &lspg_waitcryo.mutex);
}

```

#### 7.12.4.27 void lspg\_waitcryo\_cb ( lspg\_query\_queue\_t \* qqp, PGresult \* pgr )

Definition at line 497 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);
}

```

#### 7.12.4.28 void lspg\_zoom\_lut.call ( )

#### 7.12.4.29 int lspmac\_getBIPosition ( lspmac\_bi\_t \* )

get binary input value

Definition at line 1565 of file lspmac.c.

```

    {
int rtn;
pthread_mutex_lock( &bip->mutex);
rtn = bip->position;
pthread_mutex_unlock( &bip->mutex);
return rtn;
}

```

#### 7.12.4.30 double lspmac\_getPosition ( lspmac\_motor\_t \* mp )

get the motor position (with locking)

##### Parameters

<i>mp</i>	the motor object
-----------	------------------

Definition at line 1336 of file lspmac.c.

```

{
    double rtn;
    pthread_mutex_lock( &(mp->mutex));
    rtn = mp->position;
    pthread_mutex_unlock( &(mp->mutex));
    return rtn;
}

```

#### 7.12.4.31 void lspmac\_home1\_queue( lspmac\_motor\_t \* mp )

Home the motor.

##### Parameters

in	<i>mp</i>	motor we are concerned about
----	-----------	------------------------------

Definition at line 1203 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->homing) {
        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].homing || ((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->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_SockSendDPLine( mp->name, "%d$*",
        motor_num);
}

pthread_mutex_unlock( &(mp->mutex));
}

```

#### 7.12.4.32 void lspmac\_init( int, int )

Initialize this module.

Definition at line 2891 of file lspmac.c.

```

{
    md2_status_t *p;

    // Set our global harvest flags
    getivars = ivarsflag;
    getmvars = mvarsflag;

    // All important status mutex
    pthread_mutex_init( &md2_status_mutex, NULL);

    //
    // Get the MD2 initialization strings
    //
    lspmac_md2_init = lsredis_get_obj( "
        md2_pmac.init");

    //
    // 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 A", "omega",
        lspmac_moveabs_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);
    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);
    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);
    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);
    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);
}

```

```

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);
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);
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);
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);
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);
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);
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);
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);
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);

fshut = lspmac_fshut_init( &(lspmac_motors
[15]));
flight = lspmac_dac_init( &(lspmac_motors[1
6]), &p->front_dac, "M1200", "frontLight.intensity",
lspmac_movedac_queue);
blight = lspmac_dac_init( &(lspmac_motors[1
7]), &p->back_dac, "M1201", "backLight.intensity",
lspmac_movedac_queue);
fscint = lspmac_dac_init( &(lspmac_motors[1
8]), &p->scint_piezo, "M1203", "scint.focus",
lspmac_movedac_queue);

smart_mag_oo = lspmac_bo_init( &(lspmac_motors
[19]), "smartMagnet", "M1100=%d", &(md2_status.acc11c_5), 0x01
);
blight_ud = lspmac_bo_init( &(lspmac_motors
[20]), "backLight", "M1101=%d", &(md2_status.acc11c_5), 0x02
);
cryo = lspmac_bo_init( &(lspmac_motors
[21]), "cryo", "M1102=%d", &(md2_status.acc11c_5), 0x04
);
dryer = lspmac_bo_init( &(lspmac_motors
[22]), "dryer", "M1103=%d", &(md2_status.acc11c_5), 0x08
);
fluo = lspmac_bo_init( &(lspmac_motors
[23]), "fluo", "M1104=%d", &(md2_status.acc11c_5), 0x10
);
flight_oo = lspmac_soft_motor_init( &(
lspmac_motors[24]), "frontLight",
lspmac_moveabs_flight_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.acc11c_1), 0x01, "Low Pressure Air OK", "
Low Pressure Air Failed");
hp_air = lspmac_bi_init( &(lspmac_bis[
1]), &(md2_status.acc11c_1), 0x02, "High Pressure Air OK", "
High Pressure Air Failed");
cryo_switch = lspmac_bi_init( &(lspmac_bis
[ 2]), &(md2_status.acc11c_1), 0x04, "CryoSwitchChanged",
"CryoSwitchChanged");
blight_down = lspmac_bi_init( &(lspmac_bis
[ 3]), &(md2_status.acc11c_1), 0x08, "Backlight Down",
"Backlight Not Down");
blight_up = lspmac_bi_init( &(lspmac_bis

```

```

    [ 4]), &(md2_status.accl1c_1), 0x10, "Backlight Up",
    "Backlight Not Up");
cryo_back = lspmac_bi_init( &(lspmac_bis
    [ 5]), &(md2_status.accl1c_1), 0x40, "Cryo Back",
    "Cryo Not Back");
fluor_back = lspmac_bi_init( &(lspmac_bis
    [ 6]), &(md2_status.accl1c_2), 0x01, "Fluor. Det. Parked",
    "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");

//
// 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 mutexs and conditions
//

pthread_mutex_init( &pmac_queue_mutex, NULL);
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, NULL);
pthread_cond_init( &lspmac_shutter_cond, NULL);
pmacfd.fd = -1;

pthread_mutex_init( &lspmac_moving_mutex, NULL);
pthread_cond_init( &lspmac_moving_cond, NULL);

pthread_mutex_init( &lspmac_ascii_mutex, NULL);

pthread_mutex_init( &lspmac_ascii_buffers_mutex,
    NULL);

```

```

//
// 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");
}

```

#### 7.12.4.33 void lspmac\_jogabs\_queue ( lspmac\_motor\_t \*, double )

Use jog to move motor to requested position.

Definition at line 2665 of file lspmac.c.

```

{
    lspmac_move_or_jog_abs_queue( mp,
        requested_position, 1);
}

```

#### 7.12.4.34 void 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.

< 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 2515 of file lspmac.c.

```

{
    char *fmt;
    int q100;
    int requested_pos_cnts;
    int coord_num, motor_num;
    char *axis;
    double u2c;

    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);

    if( u2c == 0.0) {

```

```

    //
    // Shouldn't try moving a motor that has no units defined
    //
    pthread_mutex_unlock( &(mp->mutex));
    return;
}
mp->requested_position = requested_position;
mp->not_done = 1;
mp->motion_seen = 0;
mp->command_sent = 0;
mp->requested_pos_cnts = u2c * requested_position;
requested_pos_cnts = mp->requested_pos_cnts;

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
    // TODO: put in a timeout so we have a way out if something goes wrong
    // TODO: are we sure this thread is not the one moving it?
    //
    pthread_mutex_lock( &lspmac_moving_mutex);
    lslogging_log_message( "lspmac_moveabs_queue: waiting
        for previous moves to end. lspmac_moving_flags = %0x", lspmac_moving_flags
    );
    while( (lspmac_moving_flags & q100) != 0 )
        pthread_cond_wait( &lspmac_moving_cond, &
            lspmac_moving_mutex);
    pthread_mutex_unlock( &lspmac_moving_mutex);
    lslogging_log_message( "lspmac_moveabs_queue: Done.
        lspmac_moving_flags = %0x", lspmac_moving_flags);

    //
    // 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
    //
    pthread_mutex_lock( &lspmac_moving_mutex);
    lslogging_log_message( "lspmac_moveabs_queue: waiting

```



```

        for moving flag to propagate. lspmac_moving_flags = %0x", lspmac_moving_flags
    );
    while( (lspmac_moving_flags & q100) == 0)
        pthread_cond_wait( &lspmac_moving_cond, &
            lspmac_moving_mutex);
    pthread_mutex_unlock( &lspmac_moving_mutex);
    lslogging_log_message( "lspmac_moveabs_queue: Done.
        lspmac_moving_flags = %0x", lspmac_moving_flags);
}
pthread_mutex_lock( &(mp->mutex));
if( use_jog) {
    lspmac_SockSendDPlane( mp->name, "##d j=%d",
        motor_num, requested_pos_cnts);
} else {
    lspmac_SockSendDPlane( mp->name, fmt, coord_num,
        requested_pos_cnts, q100);
}
pthread_mutex_unlock( &(mp->mutex));

free( axis);
}

```

#### 7.12.4.35 void lspmac\_move\_or\_jog\_preset\_queue ( lspmac\_motor\_t \*, char \*, int )

move using a preset value

Definition at line 2635 of file lspmac.c.

```

{
    double pos;
    int err;

    if( preset == NULL || *preset == 0)
        return;

    err = lsredis_find_preset( mp->name, preset, &pos);

    if( err != 0)
        lspmac_move_or_jog_abs_queue( mp, pos, use_jog)
        ;
}

```

#### 7.12.4.36 void lspmac\_move\_or\_jog\_queue ( lspmac\_motor\_t \*, double, int )

#### 7.12.4.37 void lspmac\_move\_preset\_queue ( lspmac\_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>	lspmac motor pointer
<i>name</i>	Name of the preset to use

Definition at line 2284 of file lspmac.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;

    mp->moveAbs( mp, pos);
    lslogging_log_message( "lspmac_move_preset_queue: moving
        %s to preset '%s' (%f)", mp->name, preset_name, pos);
}

```

```
}

```

#### 7.12.4.38 void lspmac\_moveabs\_queue ( lspmac\_motor\_t \*, double )

Use coordinate system motion program, if available, to move motor to requested position.

Definition at line 2655 of file lspmac.c.

```

{
    lspmac_move_or_jog_abs_queue( mp,
        requested_position, 0);
}

```

#### 7.12.4.39 void lspmac\_moveabs\_wait ( lspmac\_motor\_t \* mp )

Wait for motor to finish moving.

Assume motion already queued, now just wait

##### Parameters

in	<i>mp</i>	The motor object to wait for
----	-----------	------------------------------

Definition at line 2677 of file lspmac.c.

```

{
    struct timespec wt;
    int return_code;

    //
    // Copy the queue item for the most recent move request
    //
    pthread_mutex_lock( &(mp->mutex));

    while( mp->command_sent == 0)
        pthread_cond_wait( &mp->cond, &mp->mutex);
    pthread_mutex_unlock( &(mp->mutex));

    //
    // 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.
    //
    // This sets up a one second wait
    //
    clock_gettime( CLOCK_REALTIME, &wt);
    wt.tv_sec++;

    return_code=0;

    pthread_mutex_lock( &(mp->mutex));
    while( mp->motion_seen == 0 && return_code == 0)
        return_code = pthread_cond_timedwait( &(mp->cond), &(mp->mutex), &
            wt);

    if( return_code == 0) {
        //
        // wait for the motion that we know has started to finish
        //
        while( mp->not_done)
            pthread_cond_wait( &(mp->cond), &(mp->mutex));
    }

    //
    // 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));
}

```

## 7.12.4.40 void lspmac\_run ( )

Start up the lspmac thread.

Definition at line 3263 of file lspmac.c.

```

    {
        char **inits;
        lspmac_motor_t *mp;
        char evts[64];
        int i;
        int active;

        pthread_create( &pmac_thread, NULL, lspmac_worker,
            NULL);

        lsevents_add_listener( "CryoSwitchChanged",
            lspmac_cryoSwitchChanged_cb);
        lsevents_add_listener( "scint In Position",
            lspmac_scint_inPosition_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 In Position",
            lspmac_light_zoom_cb);

        for( i=0; i<lspmac_nmotors; i++) {
            snprintf( evts, sizeof( evts)-1, "%s command done", 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( NULL, '\x18')
            ;
        {
            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);
        }

        //
        // Initialize the MD2 pmac (ie, turn on the right plcc's etc)
        //
        /*
        for( inits = lsredis_get_string_array(lspmac_md2_init); *inits != NULL;
            inits++) {
            lspmac_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<lspmac_nmotors; i++) {
            mp = &(lspmac_motors[i]);
            active = lsredis_getb( mp->active);

            // 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( "lspmac_run: motor %s is
    neither active nor inactive (!?)", mp->name);
    inits = NULL;
}
if( inits != NULL) {
    while( *inits != NULL) {
        lspmac_SockSendDpline( NULL, *inits);
        inits++;
    }
}
}
}

```

#### 7.12.4.41 void lspmac\_SockSendDpline ( char \*, char \* *fmt*, ... )

prepare (queue up) a line to send the dpram ascii command interface

Definition at line 1910 of file lspmac.c.

```

{
    va_list arg_ptr;
    uint32_t index;
    char *pl;

    pthread_mutex_lock( &lspmac_ascii_mutex);
    index = lspmac_dpascii_on++ % LSPMAC_DPASCII_QUEUE_LENGTH
    ;

    pl = lspmac_dpascii_queue[index].pl;

    va_start( arg_ptr, fmt);
    vsnprintf( pl, 159, fmt, arg_ptr);
    pl[159] = 0;
    va_end( arg_ptr);

    lspmac_dpascii_queue[index].event = event;

    pthread_mutex_unlock( &lspmac_ascii_mutex);
}

```

#### 7.12.4.42 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 1058 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( payload);
}

```

```

return lspmac_send_command( VR_DOWNLOAD,
    VR_PMAC_SENDLINE, 0, 0, strlen( payload), payload,
    lspmac_GetShortReplyCB, 0, event);
}

```

#### 7.12.4.43 void lspmac\_video\_rotate ( double secs )

Special motion program to collect centering video.

Definition at line 2486 of file lspmac.c.

```

{
double q10;          // starting position (counts)
double q11;          // delta counts
double q12;          // milliseconds to run over delta

double u2c;

if( secs <= 0.0)
    return;

omega_zero_search = 1;

pthread_mutex_lock( &(omega->mutex));
u2c = lsredis_getd( omega->u2c);

q10 = 0;
q11 = 360.0 * u2c;
q12 = 1000 * secs;

omega_zero_velocity = 360.0 * u2c / secs; //
    counts/second to back calculate zero crossing time

lspmac_SockSendDPLine( omega->name, "&1
    Q10=%.1f Q11=%.1f Q12=%.1f Q13=(I117) Q14=(I116) B240R", q10, q11, q12);
pthread_mutex_unlock( &(omega->mutex));
}

```

#### 7.12.4.44 int lsredis\_cmpnstr ( lsredis\_obj\_t \* p, char \* s, int n )

Definition at line 235 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;
}

```

#### 7.12.4.45 int lsredis\_cmpstr ( lsredis\_obj\_t \* p, char \* s )

Definition at line 224 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;
}

```

#### 7.12.4.46 int lsredis\_find\_preset ( char \* base, char \* preset\_name, double \* dval )

Definition at line 755 of file lsredis.c.

```

char s[512];
int i;
int err;
ENTRY htab_input, *htab_output;
lsredis_obj_t *p;

i = 0;
for( i=0; i<1024; i++) {
    snprintf( s, sizeof( s)-1, "%s.%s.presets.%d.name", lsredis_head
        , base, i);
    s[sizeof(s)-1] = 0;
    htab_input.key = s;
    htab_input.data = NULL;
    err = hsearch_r( htab_input, FIND, &htab_output, &lsredis_htab)
        ;
    if( err == 0) {
        // We've run out of names to look for: done
        lslogging_log_message( "lsredis_find_preset: no
            preset for motor %s named '%s'", base, preset_name);
        *dval = 0.0;
        return 0;
    }

    // Check if we have a match
    p = htab_output->data;
    if( lsredis_cmpstr( p, preset_name)==0) {
        // got a match, now look for the position
        snprintf( s, sizeof( s)-1, "%s.%s.presets.%d.position", lsredis_head
            , base, i);
        s[sizeof(s)-1] = 0;
        htab_input.key = s;
        htab_input.data = NULL;
        err = hsearch_r( htab_input, FIND, &htab_output, &lsredis_htab
            );
        if( err == 0) {
            // Name but not position? odd.
            lslogging_log_message( "lsredis_find_preset:
                Error, motor %s preset '%s' has no position defined", base, preset_name);
            *dval = 0.0;
            return 0;
        }
        p = htab_output->data;
        *dval = lsredis_getd( p);
        return 1;
    }
}
// How'd we get here?
// did someone really define that many presets? And then looked for one
// that's not there?
*dval = 0;
return 0;
}

```

#### 7.12.4.47 lsredis\_obj\_t\* lsredis\_get\_obj ( char \*, ... )

Definition at line 523 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;
}

```

#### 7.12.4.48 char\*\* lsredis\_get\_string\_array( lsredis\_obj\_t \* p )

Definition at line 364 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;
}

```

#### 7.12.4.49 int lsredis\_getb( lsredis\_obj\_t \* p )

Definition at line 377 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;
}

```

#### 7.12.4.50 double lsredis\_getd( lsredis\_obj\_t \* p )

Definition at line 338 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;
}

```

#### 7.12.4.51 long int lsredis\_getl ( lsredis\_obj\_t \* p )

Definition at line 351 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;
}

```

#### 7.12.4.52 char\* lsredis\_getstr ( lsredis\_obj\_t \* p )

return a copy of the key's string value

Definition at line 262 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;
}

```

#### 7.12.4.53 void lsredis\_init ( char \* pub, char \* re, char \* head )

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 809 of file lsredis.c.

```

{
    int err;
    int nerrmsg;
    char *errmsg;

    //
    // set up hash map to store redis objects
    //
    err = hcreate_r( 8192, &lsredis_hstab);
    if( err == 0) {
        lslogging_log_message( "lsredis_init: Cannot create
            hash table. Really bad things are going to happen. hcreate_r returned %d", err);
    }

    lsredis_head = strdup( head);
    lsredis_publisher = strdup( pub);

    pthread_mutex_init( &lsredis_mutex, NULL);
    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("10.1.0.3", 6379);
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;

err = regcomp( &lsredis_key_select_regex, re,
    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_select: %s", errmsg)
        ;
        free( errmsg);
    }
}
}

```

#### 7.12.4.54 int lsredis\_regexec ( const regex\_t \* preg, lsredis\_obj\_t \* p, size\_t nmatch, regmatch\_t \* pmatch, int eflags )

Definition at line 246 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;
}

```

#### 7.12.4.55 void lsredis\_run ( )

Definition at line 1013 of file lsredis.c.

```

    {
        pthread_create( &lsredis_thread, NULL, lsredis_worker
            , NULL);
    }

```

#### 7.12.4.56 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. Arbitrary 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 its packet (before it returns) so we should be OK with this location disappearing soon.

Definition at line 287 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);
    }

```

7.12.4.57 void `lstimer_add_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 50 of file `lstimer.c`.

```

{
    int i;
    struct timespec now;

    // Time we were called. Delay is based on call time, not queued time
    //
    clock_gettime( CLOCK_REALTIME, &now);

    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_add_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);
}

```

7.12.4.58 void `lstimer_init` ( )

Initialize the timer list and pthread stuff.

Definition at line 259 of file `lstimer.c`.

```

{
    int i;

```

```

for( i=0; i<LSTIMER_LIST_LENGTH; i++) {
    lstimer_list[i].shots = 0;
}

pthread_mutex_init( &lstimer_mutex, NULL);
pthread_cond_init( &lstimer_cond, NULL);
}

```

#### 7.12.4.59 void lstimer\_run ( )

Start up our thread.

Definition at line 273 of file lstimer.c.

```

{
    pthread_create( &lstimer_thread, NULL, lstimer_worker
, NULL);
}

```

#### 7.12.4.60 void lsupdate\_init ( )

##### 7.12.4.61 void lsupdate\_run ( )

#### 7.12.4.62 void md2cmds\_init ( )

Initialize the md2cmds module.

Definition at line 1161 of file md2cmds.c.

```

{
    memset( md2cmds_cmd, 0, sizeof( md2cmds_cmd));

    pthread_mutex_init( &md2cmds_mutex, NULL);
    pthread_cond_init( &md2cmds_cond, NULL);

    pthread_mutex_init( &md2cmds_moving_mutex, NULL);
    pthread_cond_init( &md2cmds_moving_cond, NULL);

    md2cmds_md_status_code = lsredis_get_obj
( "md2_status_code");
    lsredis_setstr( md2cmds_md_status_code, "
7");
}

```

#### 7.12.4.63 void md2cmds\_run ( )

Start up the thread.

Definition at line 1176 of file md2cmds.c.

```

{
    pthread_create( &md2cmds_thread, NULL,
md2cmds_worker, NULL);
    lsevents_add_listener( "omega crossed zero",
md2cmds_rotate_cb);
    lsevents_add_listener( ".+ (Moving|In Position)",
md2cmds_maybe_done_moving_cb);
    lsevents_add_listener( "capz (Moving|In Position)",
md2cmds_time_capz_cb);
    lsevents_add_listener( "move_prep command done",
md2cmds_move_prep_done_cb);
}

```

#### 7.12.4.64 void pgpmac\_printf ( char \* *fmt*, ... )

Terminal output routine ala printf.

##### Parameters

<i>in</i>	<i>fmt</i>	Printf style formating string
-----------	------------	-------------------------------

Definition at line 328 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);  
        wnoutrefresh( term_input);  
        doupdate();  
  
        pthread_mutex_unlock( &ncurses_mutex);  
    }
```

#### 7.12.4.65 void PmacSockSendline ( char \* *s* )

### 7.12.5 Variable Documentation

#### 7.12.5.1 lspmac\_motor\_t\* alignx

Alignment stage X.

Definition at line 88 of file lspmac.c.

#### 7.12.5.2 lspmac\_motor\_t\* aligny

Alignment stage Y.

Definition at line 89 of file lspmac.c.

#### 7.12.5.3 lspmac\_motor\_t\* alignz

Alignment stage X.

Definition at line 90 of file lspmac.c.

#### 7.12.5.4 lspmac\_motor\_t\* anal

Polaroid analyzer motor.

Definition at line 91 of file lspmac.c.

#### 7.12.5.5 lspmac\_motor\_t\* apery

Aperture Y.

Definition at line 93 of file lspmac.c.

**7.12.5.6 Ispmac\_motor\_t\* aperz**

Aperture Z.

Definition at line 94 of file Ispmac.c.

**7.12.5.7 Ispmac\_bi\_t\* arm\_parked**

(whose arm? parked where?)

Definition at line 131 of file Ispmac.c.

**7.12.5.8 Ispmac\_motor\_t\* blight**

Back Light DAC.

Definition at line 105 of file Ispmac.c.

**7.12.5.9 Ispmac\_bi\_t\* blight\_down**

Backlight is down.

Definition at line 121 of file Ispmac.c.

**7.12.5.10 Ispmac\_motor\_t\* blight\_f**

Back light scale factor.

Definition at line 114 of file Ispmac.c.

**7.12.5.11 Ispmac\_motor\_t\* blight\_ud**

Back light Up/Down actuator.

Definition at line 109 of file Ispmac.c.

**7.12.5.12 Ispmac\_bi\_t\* blight\_up**

Backlight is up.

Definition at line 122 of file Ispmac.c.

**7.12.5.13 Ispmac\_motor\_t\* capy**

Capillary Y.

Definition at line 95 of file Ispmac.c.

**7.12.5.14 Ispmac\_motor\_t\* capz**

Capillary Z.

Definition at line 96 of file Ispmac.c.

**7.12.5.15   lspmac\_motor\_t\* cenx**

Centering Table X.

Definition at line 98 of file lspmac.c.

**7.12.5.16   lspmac\_motor\_t\* ceny**

Centering Table Y.

Definition at line 99 of file lspmac.c.

**7.12.5.17   lspmac\_motor\_t\* cryo**

Move the cryostream towards or away from the crystal.

Definition at line 110 of file lspmac.c.

**7.12.5.18   lspmac\_bi\_t\* cryo\_back**

cryo is in the back position

Definition at line 123 of file lspmac.c.

**7.12.5.19   lspmac\_bi\_t\* cryo\_switch**

that little toggle switch for the cryo

Definition at line 120 of file lspmac.c.

**7.12.5.20   lspmac\_motor\_t\* dryer**

blow air on the scintillator to dry it off

Definition at line 111 of file lspmac.c.

**7.12.5.21   lspmac\_bi\_t\* etel\_init\_ok**

ETEL initialized OK.

Definition at line 128 of file lspmac.c.

**7.12.5.22   lspmac\_bi\_t\* etel\_on**

ETEL is on.

Definition at line 127 of file lspmac.c.

**7.12.5.23   lspmac\_bi\_t\* etel\_ready**

ETEL is ready.

Definition at line 126 of file lspmac.c.

**7.12.5.24 Ispmac\_motor\_t\* flight**

Front Light DAC.

Definition at line 104 of file Ispmac.c.

**7.12.5.25 Ispmac\_motor\_t\* flight\_f**

Front light scale factor.

Definition at line 115 of file Ispmac.c.

**7.12.5.26 Ispmac\_motor\_t\* flight\_oo**

Turn front light on/off.

Definition at line 113 of file Ispmac.c.

**7.12.5.27 Ispmac\_motor\_t\* fluo**

Move the fluorescence detector in/out.

Definition at line 112 of file Ispmac.c.

**7.12.5.28 Ispmac\_bi\_t\* fluor\_back**

fluor is in the back position

Definition at line 124 of file Ispmac.c.

**7.12.5.29 Ispmac\_motor\_t\* fscint**

Scintillator Piezo DAC.

Definition at line 106 of file Ispmac.c.

**7.12.5.30 Ispmac\_motor\_t\* fshut**

Fast shutter.

Definition at line 103 of file Ispmac.c.

**7.12.5.31 Ispmac\_bi\_t\* hp\_air**

High pressure air OK.

Definition at line 119 of file Ispmac.c.

**7.12.5.32 Ispmac\_motor\_t\* kappa**

Kappa.

Definition at line 100 of file Ispmac.c.



**7.12.5.33 `lspmac_bi_t*` `lp.air`**

Low pressure air OK.

Definition at line 118 of file `lspmac.c`.

**7.12.5.34 `lspg_demandairrights_t` `lspg.demandairrights`**

our demandairrights object

Definition at line 66 of file `lspg.c`.

**7.12.5.35 `lspg_getcenter_t` `lspg.getcenter`**

the getcenter object

Definition at line 65 of file `lspg.c`.

**7.12.5.36 `lspg_getcurrentsampleid_t` `lspg.getcurrentsampleid`**

our currentsample id

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

**7.12.5.37 `lspg_nextsample_t` `lspg.nextsample`**

the very next sample

Definition at line 63 of file `lspg.c`.

**7.12.5.38 `lspg_nextshot_t` `lspg.nextshot`**

the nextshot object

Definition at line 64 of file `lspg.c`.

**7.12.5.39 `lspg_starttransfer_t` `lspg.starttransfer`**

start a sample transfer

Definition at line 68 of file `lspg.c`.

**7.12.5.40 `lspg_waitcryo_t` `lspg.waitcryo`**

signal the robot

Definition at line 69 of file `lspg.c`.

**7.12.5.41 `lspmac_motor_t` `lspmac.motors[]`**

All our motors.

Definition at line 85 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 62 of file lspmac.c.

**7.12.5.43 int lspmac\_moving\_flags**

Flag used to implement motor moving condition.

Definition at line 63 of file lspmac.c.

**7.12.5.44 pthread\_mutex\_t lspmac\_moving\_mutex**

Coordinate moving motors between threads.

Definition at line 61 of file lspmac.c.

**7.12.5.45 int lspmac\_nmotors**

The number of motors we manage.

Definition at line 86 of file lspmac.c.

**7.12.5.46 pthread\_cond\_t lspmac\_shutter\_cond**

Allows waiting for the shutter status to change.

Definition at line 60 of file lspmac.c.

**7.12.5.47 int lspmac\_shutter\_has\_opened**

Indicates that the shutter had opened, perhaps briefly even if the state did not change.

Definition at line 58 of file lspmac.c.

**7.12.5.48 pthread\_mutex\_t lspmac\_shutter\_mutex**

Coordinates threads reading shutter status.

Definition at line 59 of file lspmac.c.

**7.12.5.49 int lspmac\_shutter\_state**

State of the shutter, used to detect changes.

Definition at line 57 of file lspmac.c.

**7.12.5.50 pthread\_mutex\_t md2\_status\_mutex**

Synchronize reading/writing status buffer.

Definition at line 339 of file lspmac.c.

**7.12.5.51 char md2cmds\_cmd[]**

our command;

Definition at line 20 of file md2cmds.c.

**7.12.5.52 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.

**7.12.5.53 Isredis\_obj\_t\* md2cmds\_md\_status\_code**

Definition at line 22 of file md2cmds.c.

**7.12.5.54 pthread\_mutex\_t md2cmds\_mutex**

mutex for the condition

Definition at line 11 of file md2cmds.c.

**7.12.5.55 pthread\_cond\_t md2cmds\_pg\_cond****7.12.5.56 pthread\_mutex\_t md2cmds\_pg\_mutex****7.12.5.57 lspmac\_bi\_t\* minikappa\_ok**

Minikappa is OK (whatever that means)

Definition at line 129 of file lspmac.c.

**7.12.5.58 pthread\_mutex\_t ncurses\_mutex**

allow more than one thread access to the screen

Definition at line 242 of file pgpmac.c.

**7.12.5.59 lspmac\_motor\_t\* omega**

MD2 omega axis (the air bearing)

Definition at line 87 of file lspmac.c.

**7.12.5.60 struct timespec omega\_zero\_time**

Time we believe that omega crossed zero.

Definition at line 70 of file lspmac.c.

**7.12.5.61 lspmac\_motor\_t\* phi**

Phi (not data collection axis)

Definition at line 101 of file lspmac.c.

**7.12.5.62 pthread\_cond\_t pmac\_queue\_cond**

wait for a command to be sent to PMAC before continuing

Definition at line 76 of file lspmac.c.

**7.12.5.63 pthread\_mutex\_t pmac\_queue\_mutex**

manage access to the pmac command queue

Definition at line 75 of file lspmac.c.

**7.12.5.64 lspmac\_bi\_t\* sample\_detected**

smart magnet detected sample

Definition at line 125 of file lspmac.c.

**7.12.5.65 lspmac\_motor\_t\* scint**

Scintillator Z.

Definition at line 97 of file lspmac.c.

**7.12.5.66 lspmac\_bi\_t\* shutter\_open**

shutter is open (note in pmc says this is a slow input)

Definition at line 132 of file lspmac.c.

**7.12.5.67 lspmac\_bi\_t\* smart\_mag\_err**

smart magnet error (coil broken perhaps)

Definition at line 133 of file lspmac.c.

**7.12.5.68 lspmac\_bi\_t\* smart\_mag\_off**

smart magnet is off

Definition at line 134 of file lspmac.c.

**7.12.5.69 lspmac\_bi\_t\* smart\_mag\_on**

smart magnet is on

Definition at line 130 of file lspmac.c.

**7.12.5.70 lspmac\_motor\_t\* smart\_mag\_oo**

Smart Magnet on/off.

Definition at line 108 of file lspmac.c.

**7.12.5.71 WINDOW\* term\_input**

place to put the cursor

Definition at line 238 of file pgpmac.c.

**7.12.5.72 WINDOW\* term\_output**

place to print stuff out

Definition at line 237 of file pgpmac.c.

**7.12.5.73 WINDOW\* term\_status**

shutter, lamp, air, etc status

Definition at line 239 of file pgpmac.c.

**7.12.5.74 WINDOW\* term\_status2**

shutter, lamp, air, etc status

Definition at line 240 of file pgpmac.c.

**7.12.5.75 lspmac\_motor\_t\* zoom**

Optical zoom.

Definition at line 92 of file lspmac.c.

# Index

- [\\_GNU\\_SOURCE](#)
  - [pgpmac.h, 263](#)
- [\\_\\_init\\_\\_](#)
  - [iniParser::iniParser, 16](#)
- [\\_lspmac\\_motor\\_init](#)
  - [lspmac.c, 145](#)
- [\\_lsredis\\_get\\_obj](#)
  - [lsredis.c, 207](#)
- [\\_lsredis\\_set\\_value](#)
  - [lsredis.c, 208](#)
- [acc11c\\_1](#)
  - [md2StatusStruct, 59](#)
- [acc11c\\_2](#)
  - [md2StatusStruct, 59](#)
- [acc11c\\_3](#)
  - [md2StatusStruct, 59](#)
- [acc11c\\_5](#)
  - [md2StatusStruct, 59](#)
- [acc11c\\_6](#)
  - [md2StatusStruct, 59](#)
- [active](#)
  - [lspg\\_nextshot\\_struct, 29](#)
  - [lspmac\\_motor\\_struct, 48](#)
- [active2](#)
  - [lspg\\_nextshot\\_struct, 29](#)
- [active2\\_isnull](#)
  - [lspg\\_nextshot\\_struct, 30](#)
- [active\\_init](#)
  - [lspmac\\_motor\\_struct, 48](#)
- [active\\_isnull](#)
  - [lspg\\_nextshot\\_struct, 30](#)
- [actual\\_pos\\_cnts](#)
  - [lspmac\\_motor\\_struct, 48](#)
- [actual\\_pos\\_cnts\\_p](#)
  - [lspmac\\_motor\\_struct, 48](#)
- [addRead](#)
  - [kvredis.c, 70](#)
- [addWrite](#)
  - [kvredis.c, 70](#)
- [alignx](#)
  - [lspmac.c, 194](#)
  - [pgpmac.h, 293](#)
- [alignx\\_act\\_pos](#)
  - [md2StatusStruct, 59](#)
- [alignx\\_status\\_1](#)
  - [md2StatusStruct, 59](#)
- [alignx\\_status\\_2](#)
  - [md2StatusStruct, 59](#)
- [aligny](#)
  - [lspmac.c, 195](#)
  - [pgpmac.h, 293](#)
- [aligny\\_act\\_pos](#)
  - [md2StatusStruct, 59](#)
- [aligny\\_status\\_1](#)
  - [md2StatusStruct, 59](#)
- [aligny\\_status\\_2](#)
  - [md2StatusStruct, 59](#)
- [alignz](#)
  - [lspmac.c, 195](#)
  - [pgpmac.h, 293](#)
- [alignz\\_act\\_pos](#)
  - [md2StatusStruct, 59](#)
- [alignz\\_status\\_1](#)
  - [md2StatusStruct, 59](#)
- [alignz\\_status\\_2](#)
  - [md2StatusStruct, 60](#)
- [anal](#)
  - [lspmac.c, 195](#)
  - [pgpmac.h, 293](#)
- [analyzer\\_act\\_pos](#)
  - [md2StatusStruct, 60](#)
- [analyzer\\_status\\_1](#)
  - [md2StatusStruct, 60](#)
- [analyzer\\_status\\_2](#)
  - [md2StatusStruct, 60](#)
- [aperturey\\_act\\_pos](#)
  - [md2StatusStruct, 60](#)
- [aperturey\\_status\\_1](#)
  - [md2StatusStruct, 60](#)
- [aperturey\\_status\\_2](#)
  - [md2StatusStruct, 60](#)
- [aperturez\\_act\\_pos](#)
  - [md2StatusStruct, 60](#)
- [aperturez\\_status\\_1](#)
  - [md2StatusStruct, 60](#)
- [aperturez\\_status\\_2](#)
  - [md2StatusStruct, 60](#)
- [apery](#)
  - [lspmac.c, 195](#)
  - [pgpmac.h, 293](#)
- [aperz](#)
  - [lspmac.c, 195](#)
  - [pgpmac.h, 293](#)
- [arm\\_parked](#)
  - [lspmac.c, 195](#)
  - [pgpmac.h, 294](#)
- [avalue](#)
  - [lsredis\\_obj\\_struct, 53](#)

- ax
  - lspg\_nextshot\_struct, 30
- ax2
  - lspg\_nextshot\_struct, 30
- ax2\_isnull
  - lspg\_nextshot\_struct, 30
- ax\_isnull
  - lspg\_nextshot\_struct, 30
- axis
  - lspmac\_motor\_struct, 48
- ay
  - lspg\_nextshot\_struct, 30
- ay2
  - lspg\_nextshot\_struct, 30
- ay2\_isnull
  - lspg\_nextshot\_struct, 30
- ay\_isnull
  - lspg\_nextshot\_struct, 30
- az
  - lspg\_nextshot\_struct, 30
- az2
  - lspg\_nextshot\_struct, 31
- az2\_isnull
  - lspg\_nextshot\_struct, 31
- az\_isnull
  - lspg\_nextshot\_struct, 31
- b
  - mk\_pgpmac\_redis, 12
- bData
  - tagEthernetCmd, 65
- back\_dac
  - md2StatusStruct, 60
- bi\_list
  - mk\_pgpmac\_redis, 12
- blight
  - lspmac.c, 195
  - pgpmac.h, 294
- blight\_down
  - lspmac.c, 195
  - pgpmac.h, 294
- blight\_f
  - lspmac.c, 195
  - pgpmac.h, 294
- blight\_ud
  - lspmac.c, 195
  - pgpmac.h, 294
- blight\_up
  - lspmac.c, 196
  - pgpmac.h, 294
- bvalue
  - lsredis\_obj\_struct, 53
- capv
  - lspmac.c, 196
  - pgpmac.h, 294
- capv\_act\_pos
  - md2StatusStruct, 60
- capv\_status\_1
  - md2StatusStruct, 61
- capv\_status\_2
  - md2StatusStruct, 61
- capz
  - lspmac.c, 196
  - pgpmac.h, 294
- capz\_act\_pos
  - md2StatusStruct, 61
- capz\_status\_1
  - md2StatusStruct, 61
- capz\_status\_2
  - md2StatusStruct, 61
- cb
  - lsevents\_listener\_struct, 18
- centerx\_act\_pos
  - md2StatusStruct, 61
- centerx\_status\_1
  - md2StatusStruct, 61
- centerx\_status\_2
  - md2StatusStruct, 61
- centery\_act\_pos
  - md2StatusStruct, 61
- centery\_status\_1
  - md2StatusStruct, 61
- centery\_status\_2
  - md2StatusStruct, 61
- cenx
  - lspmac.c, 196
  - pgpmac.h, 294
- ceny
  - lspmac.c, 196
  - pgpmac.h, 295
- changeEventOff
  - lspmac\_bi\_struct, 43
- changeEventOn
  - lspmac\_bi\_struct, 43
- cleanstr
  - lspmac.c, 145
- cleanup
  - kvredis.c, 70
- cmdac
  - kvredis.c, 81
- cmdfd
  - kvredis.c, 81
- command\_buf
  - lspmac\_ascii\_buffers\_struct, 42
- command\_buf\_cc
  - lspmac\_ascii\_buffers\_struct, 42
- command\_sent
  - lspmac\_motor\_struct, 48
- command\_str
  - lspmac\_ascii\_buffers\_struct, 42
- cond
  - lspg\_demandairrights\_struct, 20
  - lspg\_getcenter\_struct, 21
  - lspg\_getcurrentsampleid\_struct, 23
  - lspg\_lock\_detector\_struct, 24
  - lspg\_lock\_diffractionmeter\_struct, 25

- lspg\_nextsample\_struct, 26
- lspg\_nextshot\_struct, 31
- lspg\_seq\_run\_prep\_struct, 38
- lspg\_starttransfer\_struct, 39
- lspg\_wait\_for\_detector\_struct, 40
- lspg\_waitcryo\_struct, 40
- lspmac\_motor\_struct, 48
- lsredis\_obj\_struct, 53
- coord\_num
  - lspmac\_motor\_struct, 48
- cr\_cmd
  - lspmac.c, 196
- cryo
  - lspmac.c, 196
  - pgpmac.h, 295
- cryo\_back
  - lspmac.c, 196
  - pgpmac.h, 295
- cryo\_switch
  - lspmac.c, 196
  - pgpmac.h, 295
- cvalue
  - lsredis\_obj\_struct, 54
- cx
  - lspg\_nextshot\_struct, 31
- cx2
  - lspg\_nextshot\_struct, 31
- cx2\_isnull
  - lspg\_nextshot\_struct, 31
- cx\_isnull
  - lspg\_nextshot\_struct, 31
- cy
  - lspg\_nextshot\_struct, 31
- cy2
  - lspg\_nextshot\_struct, 31
- cy2\_isnull
  - lspg\_nextshot\_struct, 32
- cy\_isnull
  - lspg\_nextshot\_struct, 32
- dac\_mvar
  - lspmac\_motor\_struct, 48
- dax
  - lspg\_getcenter\_struct, 21
- dax\_isnull
  - lspg\_getcenter\_struct, 21
- day
  - lspg\_getcenter\_struct, 21
- day\_isnull
  - lspg\_getcenter\_struct, 22
- daz
  - lspg\_getcenter\_struct, 22
- daz\_isnull
  - lspg\_getcenter\_struct, 22
- dbmem
  - lspmac.c, 197
- dbmemIn
  - lspmac.c, 197
- dcx
  - lspg\_getcenter\_struct, 22
- dcx\_isnull
  - lspg\_getcenter\_struct, 22
- dcy
  - lspg\_getcenter\_struct, 22
- dcy\_isnull
  - lspg\_getcenter\_struct, 22
- debugCB
  - kvredis.c, 71
- delRead
  - kvredis.c, 71
- delWrite
  - kvredis.c, 71
- delay\_nsecs
  - lstimer\_list\_struct, 56
- delay\_secs
  - lstimer\_list\_struct, 56
- dryer
  - lspmac.c, 197
  - pgpmac.h, 295
- dsdir
  - lspg\_nextshot\_struct, 32
- dsdir\_isnull
  - lspg\_nextshot\_struct, 32
- dsdist
  - lspg\_nextshot\_struct, 32
- dsdist2
  - lspg\_nextshot\_struct, 32
- dsdist2\_isnull
  - lspg\_nextshot\_struct, 32
- dsdist\_isnull
  - lspg\_nextshot\_struct, 32
- dsexp
  - lspg\_nextshot\_struct, 32
- dsexp2
  - lspg\_nextshot\_struct, 32
- dsexp2\_isnull
  - lspg\_nextshot\_struct, 33
- dsexp\_isnull
  - lspg\_nextshot\_struct, 33
- dshpid
  - lspg\_nextshot\_struct, 33
- dshpid\_isnull
  - lspg\_nextshot\_struct, 33
- dskappa
  - lspg\_nextshot\_struct, 33
- dskappa2
  - lspg\_nextshot\_struct, 33
- dskappa2\_isnull
  - lspg\_nextshot\_struct, 33
- dskappa\_isnull
  - lspg\_nextshot\_struct, 33
- dsnrg
  - lspg\_nextshot\_struct, 33
- dsnrg2
  - lspg\_nextshot\_struct, 33
- dsnrg2\_isnull
  - lspg\_nextshot\_struct, 34



- dsnrg\_isnull
  - lspg\_nextshot\_struct, [34](#)
- dsomega
  - lspg\_nextshot\_struct, [34](#)
- dsomega2
  - lspg\_nextshot\_struct, [34](#)
- dsomega2\_isnull
  - lspg\_nextshot\_struct, [34](#)
- dsomega\_isnull
  - lspg\_nextshot\_struct, [34](#)
- dsoscaxis
  - lspg\_nextshot\_struct, [34](#)
- dsoscaxis2
  - lspg\_nextshot\_struct, [34](#)
- dsoscaxis2\_isnull
  - lspg\_nextshot\_struct, [34](#)
- dsoscaxis\_isnull
  - lspg\_nextshot\_struct, [34](#)
- dsowidth
  - lspg\_nextshot\_struct, [34](#)
- dsowidth2
  - lspg\_nextshot\_struct, [35](#)
- dsowidth2\_isnull
  - lspg\_nextshot\_struct, [35](#)
- dsowidth\_isnull
  - lspg\_nextshot\_struct, [35](#)
- dsphi
  - lspg\_nextshot\_struct, [35](#)
- dsphi2
  - lspg\_nextshot\_struct, [35](#)
- dsphi2\_isnull
  - lspg\_nextshot\_struct, [35](#)
- dsphi\_isnull
  - lspg\_nextshot\_struct, [35](#)
- dspid
  - lspg\_nextshot\_struct, [35](#)
- dspid\_isnull
  - lspg\_nextshot\_struct, [35](#)
- dummy1
  - md2StatusStruct, [61](#)
- dummy2
  - md2StatusStruct, [62](#)
- dummy3
  - md2StatusStruct, [62](#)
- dummy4
  - md2StatusStruct, [62](#)
- dummy5
  - md2StatusStruct, [62](#)
- dummy6
  - md2StatusStruct, [62](#)
- dummy7
  - md2StatusStruct, [62](#)
- dummy8
  - md2StatusStruct, [62](#)
- dummy9
  - md2StatusStruct, [62](#)
- dummyA
  - md2StatusStruct, [62](#)
- dummyB
  - md2StatusStruct, [62](#)
- dvalue
  - lsredis\_obj\_struct, [54](#)
- etel\_init\_ok
  - lspmac.c, [197](#)
  - pgpmac.h, [295](#)
- etel\_on
  - lspmac.c, [197](#)
  - pgpmac.h, [295](#)
- etel\_ready
  - lspmac.c, [197](#)
  - pgpmac.h, [295](#)
- ethCmdOff
  - lspmac.c, [197](#)
- ethCmdOn
  - lspmac.c, [197](#)
- ethCmdQueue
  - lspmac.c, [197](#)
- ethCmdReply
  - lspmac.c, [198](#)
- event
  - lspmac\_cmd\_queue\_struct, [45](#)
  - lspmac\_dpascii\_queue\_struct, [45](#)
  - lstimer\_list\_struct, [56](#)
- events\_name
  - lsredis\_obj\_struct, [54](#)
- evp
  - lsevents\_queue\_struct, [19](#)
- f
  - iniParser::iniParser, [17](#)
  - mk\_pgpmac\_redis, [12](#)
- fd\_service
  - kvredis.c, [72](#)
- first\_time
  - lspmac\_bi\_struct, [43](#)
- flight
  - lspmac.c, [198](#)
  - pgpmac.h, [295](#)
- flight\_f
  - lspmac.c, [198](#)
  - pgpmac.h, [296](#)
- flight\_oo
  - lspmac.c, [198](#)
  - pgpmac.h, [296](#)
- fluo
  - lspmac.c, [198](#)
  - pgpmac.h, [296](#)
- fluor\_back
  - lspmac.c, [198](#)
  - pgpmac.h, [296](#)
- fnc
  - mk\_pgpmac\_redis, [12](#)
- front\_dac
  - md2StatusStruct, [62](#)
- fs\_has\_opened
  - md2StatusStruct, [62](#)

- fs\_has\_opened\_globally
  - md2StatusStruct, 63
- fs\_is\_open
  - md2StatusStruct, 63
- fscint
  - lspmac.c, 198
  - pgpmac.h, 296
- fshut
  - lspmac.c, 198
  - pgpmac.h, 296
- gb\_cmd
  - lspmac.c, 198
- get
  - iniParser::iniParser, 16
- getcurrentsampleid
  - lspg\_getcurrentsampleid\_struct, 23
- getcurrentsampleid\_isnull
  - lspg\_getcurrentsampleid\_struct, 24
- getivars
  - lspmac.c, 199
- getmvars
  - lspmac.c, 199
- handler
  - lstimer.c, 224
- hard\_ini
  - mk\_pgpmac\_redis, 12
- hard\_ini\_fields
  - mk\_pgpmac\_redis, 12
- has\_option
  - iniParser::iniParser, 16
- has\_section
  - iniParser::iniParser, 16
- head
  - mk\_pgpmac\_redis, 12
- hex\_dump
  - lspmac.c, 146
- hi
  - mk\_pgpmac\_redis, 12
- hits
  - lsredis\_obj\_struct, 54
- home
  - lspmac\_motor\_struct, 49
- homing
  - lspmac\_motor\_struct, 49
- hp\_air
  - lspmac.c, 199
  - pgpmac.h, 296
- i
  - mk\_pgpmac\_redis, 12
- inactive\_init
  - lspmac\_motor\_struct, 49
- iniParser, 11
  - ip, 11
- iniParser.iniParser, 15
- iniParser.py, 67
- iniParser::iniParser
  - \_\_init\_\_, 16
  - f, 17
  - get, 16
  - has\_option, 16
  - has\_section, 16
  - options, 16
  - read, 16
  - sd, 17
  - sections, 17
- init\_nsecs
  - lstimer\_list\_struct, 56
- init\_secs
  - lstimer\_list\_struct, 56
- ip
  - iniParser, 11
- kappa
  - lspmac.c, 199
  - pgpmac.h, 296
- kappa\_act\_pos
  - md2StatusStruct, 63
- kappa\_status\_1
  - md2StatusStruct, 63
- kappa\_status\_2
  - md2StatusStruct, 63
- key
  - lsredis\_obj\_struct, 54
- kvredis.c, 67
  - addRead, 70
  - addWrite, 70
  - cleanup, 70
  - cmdac, 81
  - cmdfd, 81
  - debugCB, 71
  - delRead, 71
  - delWrite, 71
  - fd\_service, 72
  - kvseq, 81
  - LS\_PG\_STATE\_IDLE, 69
  - LS\_PG\_STATE\_INIT, 69
  - LS\_PG\_STATE\_RECV, 70
  - LS\_PG\_STATE\_RESET, 70
  - LS\_PG\_STATE\_SEND, 70
  - ls\_pg\_state, 81
  - lspg\_allkvs\_cb, 72
  - lspg\_connectPoll\_response, 82
  - lspg\_flush, 73
  - lspg\_next\_state, 73
  - lspg\_notice\_processor, 74
  - lspg\_pg\_connect, 74
  - lspg\_pg\_service, 75
  - lspg\_query\_next, 76
  - lspg\_query\_push, 77
  - lspg\_query\_queue, 82
  - lspg\_query\_queue\_off, 82
  - lspg\_query\_queue\_on, 82
  - lspg\_query\_queue\_reply, 82
  - lspg\_query\_queue\_t, 70
  - lspg\_query\_reply\_next, 77

- lspg\_query\_reply\_peek, 78
- lspg\_receive, 78
- lspg\_resetPoll\_response, 82
- lspg\_send\_next\_query, 79
- lspgfd, 82
- main, 79
- now, 82
- q, 82
- redisDisconnectCB, 81
- subac, 83
- subfd, 83
- kvseq
  - kvredis.c, 81
- LS\_PG\_STATE\_IDLE
  - kvredis.c, 69
  - lspg.c, 98
- LS\_PG\_STATE\_INIT
  - kvredis.c, 69
  - lspg.c, 99
- LS\_PG\_STATE\_RECV
  - kvredis.c, 70
  - lspg.c, 99
- LS\_PG\_STATE\_RESET
  - kvredis.c, 70
  - lspg.c, 99
- LS\_PG\_STATE\_SEND
  - kvredis.c, 70
  - lspg.c, 99
- LS\_PMAC\_STATE\_CR
  - lspmac.c, 141
- LS\_PMAC\_STATE\_GB
  - lspmac.c, 141
- LS\_PMAC\_STATE\_GMR
  - lspmac.c, 141
- LS\_PMAC\_STATE\_IDLE
  - lspmac.c, 142
- LS\_PMAC\_STATE\_RESET
  - lspmac.c, 142
- LS\_PMAC\_STATE\_RR
  - lspmac.c, 142
- LS\_PMAC\_STATE\_SC
  - lspmac.c, 142
- LS\_PMAC\_STATE\_WACK
  - lspmac.c, 142
- LS\_PMAC\_STATE\_WCR
  - lspmac.c, 142
- LS\_PMAC\_STATE\_WGB
  - lspmac.c, 142
- LSLOGGING\_FILE\_NAME
  - lslogging.c, 90
- LSPMAC\_PRESET\_REGEX
  - lspmac.c, 142
- LSTIMER\_LIST\_LENGTH
  - lstimer.c, 224
- last\_nsecs
  - lstimer\_list\_struct, 56
- last\_secs
  - lstimer\_list\_struct, 56
- lmsg
  - lslogging\_queue\_struct, 19
- lp\_air
  - lspmac.c, 199
  - pgpmac.h, 296
- ls\_pg\_state
  - kvredis.c, 81
  - lspg.c, 130
- ls\_pmac\_state
  - lspmac.c, 199
- lsConnect
  - lspmac.c, 146
- lsevents.c, 83
  - lsevents\_add\_listener, 85
  - lsevents\_init, 85
  - lsevents\_listener\_mutex, 88
  - lsevents\_listener\_t, 85
  - lsevents\_listeners\_p, 88
  - lsevents\_queue, 88
  - lsevents\_queue\_cond, 88
  - lsevents\_queue\_mutex, 88
  - lsevents\_queue\_off, 88
  - lsevents\_queue\_on, 89
  - lsevents\_queue\_t, 85
  - lsevents\_remove\_listener, 86
  - lsevents\_run, 86
  - lsevents\_send\_event, 86
  - lsevents\_thread, 89
  - lsevents\_worker, 87
- lsevents\_add\_listener
  - lsevents.c, 85
  - pgpmac.h, 265
- lsevents\_init
  - lsevents.c, 85
  - pgpmac.h, 265
- lsevents\_listener\_mutex
  - lsevents.c, 88
- lsevents\_listener\_struct, 17
  - cb, 18
  - next, 18
  - raw\_regexp, 18
  - re, 18
- lsevents\_listener\_t
  - lsevents.c, 85
- lsevents\_listeners\_p
  - lsevents.c, 88
- lsevents\_queue
  - lsevents.c, 88
- lsevents\_queue\_cond
  - lsevents.c, 88
- lsevents\_queue\_mutex
  - lsevents.c, 88
- lsevents\_queue\_off
  - lsevents.c, 88
- lsevents\_queue\_on
  - lsevents.c, 89
- lsevents\_queue\_struct, 18
  - evp, 19

- lsevents\_queue\_t
  - lsevents.c, 85
- lsevents\_remove\_listener
  - lsevents.c, 86
  - pgpmac.h, 266
- lsevents\_run
  - lsevents.c, 86
  - pgpmac.h, 266
- lsevents\_send\_event
  - lsevents.c, 86
  - pgpmac.h, 266
- lsevents\_thread
  - lsevents.c, 89
- lsevents\_worker
  - lsevents.c, 87
- lslogging.c, 89
  - LSLOGGING\_FILE\_NAME, 90
  - lslogging\_cond, 92
  - lslogging\_file, 92
  - lslogging\_init, 91
  - lslogging\_log\_message, 91
  - lslogging\_mutex, 92
  - lslogging\_off, 93
  - lslogging\_on, 93
  - lslogging\_queue, 93
  - lslogging\_queue\_t, 91
  - lslogging\_run, 92
  - lslogging\_thread, 93
  - lslogging\_worker, 92
- lslogging\_cond
  - lslogging.c, 92
- lslogging\_file
  - lslogging.c, 92
- lslogging\_init
  - lslogging.c, 91
  - pgpmac.h, 267
- lslogging\_log\_message
  - lslogging.c, 91
  - pgpmac.h, 267
- lslogging\_mutex
  - lslogging.c, 92
- lslogging\_off
  - lslogging.c, 93
- lslogging\_on
  - lslogging.c, 93
- lslogging\_queue
  - lslogging.c, 93
- lslogging\_queue\_struct, 19
  - lmsg, 19
  - ltime, 19
- lslogging\_queue\_t
  - lslogging.c, 91
- lslogging\_run
  - lslogging.c, 92
  - pgpmac.h, 268
- lslogging\_thread
  - lslogging.c, 93
- lslogging\_worker
  - lslogging.c, 92
- lslogging.c, 92
  - LS\_PG\_STATE\_IDLE, 98
  - LS\_PG\_STATE\_INIT, 99
  - LS\_PG\_STATE\_RECV, 99
  - LS\_PG\_STATE\_RESET, 99
  - LS\_PG\_STATE\_SEND, 99
  - ls\_pg\_state, 130
  - lspg\_array2ptrs, 100
  - lspg\_cmd\_cb, 101
  - lspg\_connectPoll\_response, 130
  - lspg\_demandairrights, 130
  - lspg\_demandairrights\_all, 101
  - lspg\_demandairrights\_call, 102
  - lspg\_demandairrights\_cb, 102
  - lspg\_demandairrights\_init, 102
  - lspg\_demandairrights\_wait, 102
  - lspg\_flush, 102
  - lspg\_getcenter, 130
  - lspg\_getcenter\_all, 103
  - lspg\_getcenter\_call, 103
  - lspg\_getcenter\_cb, 103
  - lspg\_getcenter\_done, 104
  - lspg\_getcenter\_init, 104
  - lspg\_getcenter\_wait, 105
  - lspg\_getcurrentsampleid, 130
  - lspg\_getcurrentsampleid\_call, 105
  - lspg\_getcurrentsampleid\_cb, 105
  - lspg\_getcurrentsampleid\_init, 105
  - lspg\_getcurrentsampleid\_read, 106
  - lspg\_getcurrentsampleid\_wait\_for\_id, 106
  - lspg\_init, 106
  - lspg\_lock\_detector, 130
  - lspg\_lock\_detector\_all, 107
  - lspg\_lock\_detector\_call, 107
  - lspg\_lock\_detector\_cb, 107
  - lspg\_lock\_detector\_done, 107
  - lspg\_lock\_detector\_init, 107
  - lspg\_lock\_detector\_t, 99
  - lspg\_lock\_detector\_wait, 108
  - lspg\_lock\_diffractionmeter, 131
  - lspg\_lock\_diffractionmeter\_all, 108
  - lspg\_lock\_diffractionmeter\_call, 108
  - lspg\_lock\_diffractionmeter\_cb, 108
  - lspg\_lock\_diffractionmeter\_done, 109
  - lspg\_lock\_diffractionmeter\_init, 109
  - lspg\_lock\_diffractionmeter\_t, 99
  - lspg\_lock\_diffractionmeter\_wait, 109
  - lspg\_next\_state, 109
  - lspg\_nextaction\_cb, 110
  - lspg\_nextsample, 131
  - lspg\_nextsample\_all, 111
  - lspg\_nextsample\_call, 111
  - lspg\_nextsample\_cb, 111
  - lspg\_nextsample\_done, 112
  - lspg\_nextsample\_init, 112
  - lspg\_nextsample\_wait, 112
  - lspg\_nextshot, 131

- lspg\_nextshot\_call, 112
- lspg\_nextshot\_cb, 113
- lspg\_nextshot\_done, 116
- lspg\_nextshot\_init, 117
- lspg\_nextshot\_wait, 117
- lspg\_notice\_processor, 117
- lspg\_pg\_connect, 117
- lspg\_pg\_service, 118
- lspg\_query\_next, 120
- lspg\_query\_push, 120
- lspg\_query\_queue, 131
- lspg\_query\_queue\_off, 131
- lspg\_query\_queue\_on, 131
- lspg\_query\_queue\_reply, 131
- lspg\_query\_reply\_next, 121
- lspg\_query\_reply\_peek, 121
- lspg\_queue\_cond, 131
- lspg\_queue\_mutex, 131
- lspg\_receive, 121
- lspg\_resetPoll\_response, 132
- lspg\_run, 122
- lspg\_send\_next\_query, 122
- lspg\_seq\_run\_prep, 132
- lspg\_seq\_run\_prep\_all, 123
- lspg\_seq\_run\_prep\_call, 123
- lspg\_seq\_run\_prep\_cb, 124
- lspg\_seq\_run\_prep\_done, 124
- lspg\_seq\_run\_prep\_init, 124
- lspg\_seq\_run\_prep\_t, 99
- lspg\_seq\_run\_prep\_wait, 125
- lspg\_sig\_service, 125
- lspg\_starttransfer, 132
- lspg\_starttransfer\_all, 125
- lspg\_starttransfer\_call, 126
- lspg\_starttransfer\_cb, 126
- lspg\_starttransfer\_done, 126
- lspg\_starttransfer\_init, 126
- lspg\_starttransfer\_wait, 126
- lspg\_thread, 132
- lspg\_wait\_for\_detector, 132
- lspg\_wait\_for\_detector\_all, 127
- lspg\_wait\_for\_detector\_call, 127
- lspg\_wait\_for\_detector\_cb, 127
- lspg\_wait\_for\_detector\_done, 127
- lspg\_wait\_for\_detector\_init, 128
- lspg\_wait\_for\_detector\_t, 99
- lspg\_wait\_for\_detector\_wait, 128
- lspg\_waitcryo, 132
- lspg\_waitcryo\_all, 128
- lspg\_waitcryo\_cb, 128
- lspg\_waitcryo\_init, 128
- lspg\_worker, 129
- lspgfd, 132
- lspmac\_sample\_detector\_cb, 130
- now, 132
- q, 132
- lspg\_allkvs\_cb
  - kvredis.c, 72
- lspg\_array2ptrs
  - lspg.c, 100
  - pgpmac.h, 268
- lspg\_cmd\_cb
  - lspg.c, 101
- lspg\_connectPoll\_response
  - kvredis.c, 82
  - lspg.c, 130
- lspg\_demandairrights
  - lspg.c, 130
  - pgpmac.h, 297
- lspg\_demandairrights\_all
  - lspg.c, 101
  - pgpmac.h, 269
- lspg\_demandairrights\_call
  - lspg.c, 102
- lspg\_demandairrights\_cb
  - lspg.c, 102
- lspg\_demandairrights\_init
  - lspg.c, 102
- lspg\_demandairrights\_struct, 20
  - cond, 20
  - mutex, 20
  - new\_value\_ready, 20
- lspg\_demandairrights\_t
  - pgpmac.h, 264
- lspg\_demandairrights\_wait
  - lspg.c, 102
- lspg\_flush
  - kvredis.c, 73
  - lspg.c, 102
- lspg\_getcenter
  - lspg.c, 130
  - pgpmac.h, 297
- lspg\_getcenter\_all
  - lspg.c, 103
- lspg\_getcenter\_call
  - lspg.c, 103
  - pgpmac.h, 270
- lspg\_getcenter\_cb
  - lspg.c, 103
- lspg\_getcenter\_done
  - lspg.c, 104
  - pgpmac.h, 270
- lspg\_getcenter\_init
  - lspg.c, 104
- lspg\_getcenter\_struct, 20
  - cond, 21
  - dax, 21
  - dax\_isnull, 21
  - day, 21
  - day\_isnull, 22
  - daz, 22
  - daz\_isnull, 22
  - dcx, 22
  - dcx\_isnull, 22
  - dcy, 22
  - dcy\_isnull, 22

- mutex, 22
- new\_value\_ready, 22
- no\_rows\_returned, 22
- zoom, 23
- zoom\_isnull, 23
- lspg\_getcenter\_t
  - pgpmac.h, 264
- lspg\_getcenter\_wait
  - lspg.c, 105
  - pgpmac.h, 270
- lspg\_getcurrentsampleid
  - lspg.c, 130
  - pgpmac.h, 297
- lspg\_getcurrentsampleid\_call
  - lspg.c, 105
- lspg\_getcurrentsampleid\_cb
  - lspg.c, 105
- lspg\_getcurrentsampleid\_init
  - lspg.c, 105
- lspg\_getcurrentsampleid\_read
  - lspg.c, 106
- lspg\_getcurrentsampleid\_struct, 23
  - cond, 23
  - getcurrentsampleid, 23
  - getcurrentsampleid\_isnull, 24
  - mutex, 24
  - new\_value\_ready, 24
  - no\_rows\_returned, 24
- lspg\_getcurrentsampleid\_t
  - pgpmac.h, 264
- lspg\_getcurrentsampleid\_wait\_for\_id
  - lspg.c, 106
  - pgpmac.h, 270
- lspg\_init
  - lspg.c, 106
  - pgpmac.h, 270
- lspg\_lock\_detector
  - lspg.c, 130
- lspg\_lock\_detector\_all
  - lspg.c, 107
- lspg\_lock\_detector\_call
  - lspg.c, 107
- lspg\_lock\_detector\_cb
  - lspg.c, 107
- lspg\_lock\_detector\_done
  - lspg.c, 107
- lspg\_lock\_detector\_init
  - lspg.c, 107
- lspg\_lock\_detector\_struct, 24
  - cond, 24
  - mutex, 24
  - new\_value\_ready, 25
- lspg\_lock\_detector\_t
  - lspg.c, 99
- lspg\_lock\_detector\_wait
  - lspg.c, 108
- lspg\_lock\_diffractionmeter
  - lspg.c, 131
- lspg\_lock\_diffractionmeter\_all
  - lspg.c, 108
- lspg\_lock\_diffractionmeter\_call
  - lspg.c, 108
- lspg\_lock\_diffractionmeter\_cb
  - lspg.c, 108
- lspg\_lock\_diffractionmeter\_done
  - lspg.c, 109
- lspg\_lock\_diffractionmeter\_init
  - lspg.c, 109
- lspg\_lock\_diffractionmeter\_struct, 25
  - cond, 25
  - mutex, 25
  - new\_value\_ready, 25
- lspg\_lock\_diffractionmeter\_t
  - lspg.c, 99
- lspg\_lock\_diffractionmeter\_wait
  - lspg.c, 109
- lspg\_next\_state
  - kvredis.c, 73
  - lspg.c, 109
- lspg\_nextaction\_cb
  - lspg.c, 110
- lspg\_nextsample
  - lspg.c, 131
  - pgpmac.h, 297
- lspg\_nextsample\_all
  - lspg.c, 111
  - pgpmac.h, 271
- lspg\_nextsample\_call
  - lspg.c, 111
- lspg\_nextsample\_cb
  - lspg.c, 111
- lspg\_nextsample\_done
  - lspg.c, 112
- lspg\_nextsample\_init
  - lspg.c, 112
- lspg\_nextsample\_struct, 25
  - cond, 26
  - mutex, 26
  - new\_value\_ready, 26
  - nextsample, 26
  - nextsample\_isnull, 26
  - no\_rows\_returned, 26
- lspg\_nextsample\_t
  - pgpmac.h, 264
- lspg\_nextsample\_wait
  - lspg.c, 112
- lspg\_nextshot
  - lspg.c, 131
  - pgpmac.h, 297
- lspg\_nextshot\_call
  - lspg.c, 112
  - pgpmac.h, 271
- lspg\_nextshot\_cb
  - lspg.c, 113
- lspg\_nextshot\_done
  - lspg.c, 116

- pgpmac.h, 271
- lspg\_nextshot\_init
  - lspg.c, 117
- lspg\_nextshot\_struct, 27
  - active, 29
  - active2, 29
  - active2\_isnull, 30
  - active\_isnull, 30
  - ax, 30
  - ax2, 30
  - ax2\_isnull, 30
  - ax\_isnull, 30
  - ay, 30
  - ay2, 30
  - ay2\_isnull, 30
  - ay\_isnull, 30
  - az, 30
  - az2, 31
  - az2\_isnull, 31
  - az\_isnull, 31
  - cond, 31
  - cx, 31
  - cx2, 31
  - cx2\_isnull, 31
  - cx\_isnull, 31
  - cy, 31
  - cy2, 31
  - cy2\_isnull, 32
  - cy\_isnull, 32
  - dsdir, 32
  - dsdir\_isnull, 32
  - dsdist, 32
  - dsdist2, 32
  - dsdist2\_isnull, 32
  - dsdist\_isnull, 32
  - dsexp, 32
  - dsexp2, 32
  - dsexp2\_isnull, 33
  - dsexp\_isnull, 33
  - dshpid, 33
  - dshpid\_isnull, 33
  - dskappa, 33
  - dskappa2, 33
  - dskappa2\_isnull, 33
  - dskappa\_isnull, 33
  - dsnrg, 33
  - dsnrg2, 33
  - dsnrg2\_isnull, 34
  - dsnrg\_isnull, 34
  - dsomega, 34
  - dsomega2, 34
  - dsomega2\_isnull, 34
  - dsomega\_isnull, 34
  - dsoscaxis, 34
  - dsoscaxis2, 34
  - dsoscaxis2\_isnull, 34
  - dsoscaxis\_isnull, 34
  - dsowidth, 34
  - dsowidth2, 35
  - dsowidth2\_isnull, 35
  - dsowidth\_isnull, 35
  - dsphi, 35
  - dsphi2, 35
  - dsphi2\_isnull, 35
  - dsphi\_isnull, 35
  - dspid, 35
  - dspid\_isnull, 35
  - mutex, 35
  - new\_value\_ready, 36
  - no\_rows\_returned, 36
  - sfn, 36
  - sfn\_isnull, 36
  - sindex, 36
  - sindex2, 36
  - sindex2\_isnull, 36
  - sindex\_isnull, 36
  - skey, 36
  - skey\_isnull, 36
  - sstart, 37
  - sstart2, 37
  - sstart2\_isnull, 37
  - sstart\_isnull, 37
  - stype, 37
  - stype2, 37
  - stype2\_isnull, 37
  - stype\_isnull, 37
- lspg\_nextshot\_t
  - pgpmac.h, 264
- lspg\_nextshot\_wait
  - lspg.c, 117
  - pgpmac.h, 272
- lspg\_notice\_processor
  - kvredis.c, 74
  - lspg.c, 117
- lspg\_pg\_connect
  - kvredis.c, 74
  - lspg.c, 117
- lspg\_pg\_service
  - kvredis.c, 75
  - lspg.c, 118
- lspg\_query\_next
  - kvredis.c, 76
  - lspg.c, 120
- lspg\_query\_push
  - kvredis.c, 77
  - lspg.c, 120
  - pgpmac.h, 272
- lspg\_query\_queue
  - kvredis.c, 82
  - lspg.c, 131
- lspg\_query\_queue\_off
  - kvredis.c, 82
  - lspg.c, 131
- lspg\_query\_queue\_on
  - kvredis.c, 82
  - lspg.c, 131

- lspg\_query\_queue\_reply
  - kvredis.c, [82](#)
  - lspg.c, [131](#)
- lspg\_query\_queue\_t
  - kvredis.c, [70](#)
  - pgpmac.h, [264](#)
- lspg\_query\_reply\_next
  - kvredis.c, [77](#)
  - lspg.c, [121](#)
- lspg\_query\_reply\_peek
  - kvredis.c, [78](#)
  - lspg.c, [121](#)
- lspg\_queue\_cond
  - lspg.c, [131](#)
- lspg\_queue\_mutex
  - lspg.c, [131](#)
- lspg\_receive
  - kvredis.c, [78](#)
  - lspg.c, [121](#)
- lspg\_resetPoll\_response
  - kvredis.c, [82](#)
  - lspg.c, [132](#)
- lspg\_run
  - lspg.c, [122](#)
  - pgpmac.h, [272](#)
- lspg\_send\_next\_query
  - kvredis.c, [79](#)
  - lspg.c, [122](#)
- lspg\_seq\_run\_prep
  - lspg.c, [132](#)
- lspg\_seq\_run\_prep\_all
  - lspg.c, [123](#)
  - pgpmac.h, [273](#)
- lspg\_seq\_run\_prep\_call
  - lspg.c, [123](#)
- lspg\_seq\_run\_prep\_cb
  - lspg.c, [124](#)
- lspg\_seq\_run\_prep\_done
  - lspg.c, [124](#)
- lspg\_seq\_run\_prep\_init
  - lspg.c, [124](#)
- lspg\_seq\_run\_prep\_struct, [37](#)
  - cond, [38](#)
  - mutex, [38](#)
  - new\_value\_ready, [38](#)
- lspg\_seq\_run\_prep\_t
  - lspg.c, [99](#)
- lspg\_seq\_run\_prep\_wait
  - lspg.c, [125](#)
- lspg\_sig\_service
  - lspg.c, [125](#)
- lspg\_starttransfer
  - lspg.c, [132](#)
  - pgpmac.h, [297](#)
- lspg\_starttransfer\_all
  - lspg.c, [125](#)
- lspg\_starttransfer\_call
  - lspg.c, [126](#)
- pgpmac.h, [273](#)
- lspg\_starttransfer\_cb
  - lspg.c, [126](#)
- lspg\_starttransfer\_done
  - lspg.c, [126](#)
  - pgpmac.h, [273](#)
- lspg\_starttransfer\_init
  - lspg.c, [126](#)
- lspg\_starttransfer\_struct, [38](#)
  - cond, [39](#)
  - mutex, [39](#)
  - new\_value\_ready, [39](#)
  - no\_rows\_returned, [39](#)
  - starttransfer, [39](#)
- lspg\_starttransfer\_t
  - pgpmac.h, [264](#)
- lspg\_starttransfer\_wait
  - lspg.c, [126](#)
  - pgpmac.h, [273](#)
- lspg\_thread
  - lspg.c, [132](#)
- 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, [39](#)
  - cond, [40](#)
  - mutex, [40](#)
  - new\_value\_ready, [40](#)
- lspg\_wait\_for\_detector\_t
  - lspg.c, [99](#)
- lspg\_wait\_for\_detector\_wait
  - lspg.c, [128](#)
- lspg\_waitcryo
  - lspg.c, [132](#)
  - pgpmac.h, [297](#)
- lspg\_waitcryo\_all
  - lspg.c, [128](#)
  - pgpmac.h, [274](#)
- lspg\_waitcryo\_cb
  - lspg.c, [128](#)
  - pgpmac.h, [274](#)
- lspg\_waitcryo\_init
  - lspg.c, [128](#)
- lspg\_waitcryo\_struct, [40](#)
  - cond, [40](#)
  - mutex, [40](#)
  - new\_value\_ready, [41](#)
- lspg\_waitcryo\_t
  - pgpmac.h, [264](#)



lspg\_worker  
   lspg.c, 129  
 lspg\_zoom\_lut\_call  
   pgpmac.h, 274  
 lspgQueryQueueStruct, 41  
   onResponse, 41  
   qs, 41  
 lspgfd  
   kvredis.c, 82  
   lspg.c, 132  
 lspmac.c, 133  
   \_lspmac\_motor\_init, 145  
   alignx, 194  
   aligny, 195  
   alignz, 195  
   anal, 195  
   apery, 195  
   aperz, 195  
   arm\_parked, 195  
   blight, 195  
   blight\_down, 195  
   blight\_f, 195  
   blight\_ud, 195  
   blight\_up, 196  
   copy, 196  
   capz, 196  
   cenx, 196  
   ceny, 196  
   cleanstr, 145  
   cr\_cmd, 196  
   cryo, 196  
   cryo\_back, 196  
   cryo\_switch, 196  
   dbmem, 197  
   dbmemIn, 197  
   dryer, 197  
   etel\_init\_ok, 197  
   etel\_on, 197  
   etel\_ready, 197  
   ethCmdOff, 197  
   ethCmdOn, 197  
   ethCmdQueue, 197  
   ethCmdReply, 198  
   flight, 198  
   flight\_f, 198  
   flight\_oo, 198  
   fluo, 198  
   fluor\_back, 198  
   fscint, 198  
   fshut, 198  
   gb\_cmd, 198  
   getivars, 199  
   getmvars, 199  
   hex\_dump, 146  
   hp\_air, 199  
   kappa, 199  
   LS\_PMAC\_STATE\_CR, 141  
   LS\_PMAC\_STATE\_GB, 141  
   LS\_PMAC\_STATE\_GMR, 141  
   LS\_PMAC\_STATE\_IDLE, 142  
   LS\_PMAC\_STATE\_RR, 142  
   LS\_PMAC\_STATE\_SC, 142  
   LS\_PMAC\_STATE\_WACK, 142  
   LS\_PMAC\_STATE\_WCR, 142  
   LS\_PMAC\_STATE\_WGB, 142  
   LSPMAC\_PRESET\_REGEX, 142  
   lp\_air, 199  
   ls\_pmac\_state, 199  
   lsConnect, 146  
   lspmac\_Error, 151  
   lspmac\_GetAllVars, 158  
   lspmac\_GetAllVarsCB, 158  
   lspmac\_GetAllMVars, 158  
   lspmac\_GetAllMVarsCB, 158  
   lspmac\_GetShortReplyCB, 160  
   lspmac\_Getmem, 159  
   lspmac\_GetmemReplyCB, 159  
   lspmac\_Reset, 181  
   lspmac\_SendControlReplyPrintCB, 186  
   lspmac\_Service, 186  
   lspmac\_SockFlush, 190  
   lspmac\_SockGetmem, 190  
   lspmac\_SockSendControlCharPrint, 190  
   lspmac\_SockSendDPLine, 190  
   lspmac\_SockSendDPQueue, 191  
   lspmac\_SockSendline, 191  
   lspmac\_SockSendline\_nr, 192  
   lspmac\_ascii\_buffers, 199  
   lspmac\_ascii\_buffers\_mutex, 199  
   lspmac\_ascii\_buffers\_t, 145  
   lspmac\_ascii\_busy, 199  
   lspmac\_ascii\_mutex, 199  
   lspmac\_asciicmdCB, 147  
   lspmac\_backLight\_down\_cb, 147  
   lspmac\_backLight\_up\_cb, 148  
   lspmac\_bi\_init, 148  
   lspmac\_bis, 200  
   lspmac\_blight\_lut\_setup, 148  
   lspmac\_bo\_init, 149  
   lspmac\_bo\_read, 149  
   lspmac\_command\_done\_cb, 150  
   lspmac\_cryoSwitchChanged\_cb, 150  
   lspmac\_dac\_init, 150  
   lspmac\_dac\_read, 151  
   lspmac\_dpascii\_off, 200  
   lspmac\_dpascii\_on, 200  
   lspmac\_dpascii\_queue, 200  
   lspmac\_dpascii\_queue\_t, 145  
   lspmac\_flight\_lut\_setup, 152  
   lspmac\_fscint\_lut\_setup, 152  
   lspmac\_fshut\_init, 153  
   lspmac\_get\_ascii, 153  
   lspmac\_get\_ascii\_cb, 153  
   lspmac\_get\_status, 155  
   lspmac\_get\_status\_cb, 155  
   lspmac\_getBIPosition, 159

- lspmac\_getPosition, 160
- lspmac\_home1\_queue, 160
- lspmac\_home2\_queue, 162
- lspmac\_init, 162
- lspmac\_jogabs\_queue, 165
- lspmac\_light\_zoom\_cb, 166
- lspmac\_lut, 166
- lspmac\_md2\_init, 200
- lspmac\_more\_ascii\_cb, 167
- lspmac\_motor\_init, 167
- lspmac\_motors, 200
- lspmac\_move\_or\_jog\_abs\_queue, 168
- lspmac\_move\_or\_jog\_preset\_queue, 170
- lspmac\_move\_preset\_queue, 170
- lspmac\_moveabs\_blight\_factor\_queue, 170
- lspmac\_moveabs\_bo\_queue, 171
- lspmac\_moveabs\_flight\_factor\_queue, 171
- lspmac\_moveabs\_frontlight\_oo\_queue, 172
- lspmac\_moveabs\_fshut\_queue, 172
- lspmac\_moveabs\_queue, 172
- lspmac\_moveabs\_timed\_queue, 173
- lspmac\_moveabs\_wait, 174
- lspmac\_movedac\_queue, 174
- lspmac\_movezoom\_queue, 175
- lspmac\_moving\_cond, 200
- lspmac\_moving\_flags, 200
- lspmac\_moving\_mutex, 200
- lspmac\_nbis, 200
- lspmac\_next\_state, 175
- lspmac\_nmotors, 201
- lspmac\_pmacmotor\_read, 177
- lspmac\_pop\_queue, 180
- lspmac\_pop\_reply, 180
- lspmac\_push\_queue, 181
- lspmac\_reset\_queue, 181
- lspmac\_rlut, 181
- lspmac\_run, 182
- lspmac\_scint\_dried\_cb, 184
- lspmac\_scint\_inPosition\_cb, 184
- lspmac\_send\_command, 184
- lspmac\_sendcmd, 185
- lspmac\_sendcmd\_nocb, 186
- lspmac\_shutter\_cond, 201
- lspmac\_shutter\_has\_opened, 201
- lspmac\_shutter\_mutex, 201
- lspmac\_shutter\_read, 189
- lspmac\_shutter\_state, 201
- lspmac\_soft\_motor\_init, 192
- lspmac\_soft\_motor\_read, 192
- lspmac\_status\_last\_time, 201
- lspmac\_status\_time, 201
- lspmac\_test\_preset, 192
- lspmac\_video\_rotate, 193
- lspmac\_worker, 193
- lspmac\_zoom\_lut\_setup, 194
- md2\_status, 201
- md2\_status\_mutex, 201
- md2\_status\_t, 145
- minikappa\_ok, 202
- now, 202
- omega, 202
- omega\_zero\_search, 202
- omega\_zero\_time, 202
- omega\_zero\_velocity, 202
- PMAC\_MIN\_CMD\_TIME, 143
- PMACPORT, 143
- phi, 202
- pmac\_cmd\_size, 143
- pmac\_error\_strs, 202
- pmac\_queue\_cond, 203
- pmac\_queue\_mutex, 203
- pmac\_thread, 203
- pmacfd, 203
- rr\_cmd, 203
- sample\_detected, 203
- scint, 203
- shutter\_open, 204
- smart\_mag\_err, 204
- smart\_mag\_off, 204
- smart\_mag\_on, 204
- smart\_mag\_oo, 204
- VR\_CTRL\_RESPONSE, 143
- VR\_DOWNLOAD, 143
- VR\_FWDOWNLOAD, 143
- VR\_IPADDRESS, 143
- VR\_PMAC\_FLUSH, 143
- VR\_PMAC\_GETBUFFER, 143
- VR\_PMAC\_GETLINE, 143
- VR\_PMAC\_GETMEM, 144
- VR\_PMAC\_GETRESPONSE, 144
- VR\_PMAC\_PORT, 144
- VR\_PMAC\_READREADY, 144
- VR\_PMAC\_SENDLINE, 144
- VR\_PMAC\_SETBIT, 144
- VR\_PMAC\_SETBITS, 144
- VR\_PMAC\_SETMEM, 144
- VR\_PMAC\_WRITEBUFFER, 144
- VR\_PMAC\_WRITEERROR, 144
- VR\_UPLOAD, 144
- zoom, 204
- lspmac\_Error
  - lspmac.c, 151
- lspmac\_GetAllIVars
  - lspmac.c, 158
- lspmac\_GetAllIVarsCB
  - lspmac.c, 158
- lspmac\_GetAllMVars
  - lspmac.c, 158
- lspmac\_GetAllMVarsCB
  - lspmac.c, 158
- lspmac\_GetShortReplyCB
  - lspmac.c, 160
- lspmac\_Getmem
  - lspmac.c, 159
- lspmac\_GetmemReplyCB
  - lspmac.c, 159

`Ispmac_Reset`  
    `Ispmac.c`, [181](#)

`Ispmac_SendControlReplyPrintCB`  
    `Ispmac.c`, [186](#)

`Ispmac_Service`  
    `Ispmac.c`, [186](#)

`Ispmac_SockFlush`  
    `Ispmac.c`, [190](#)

`Ispmac_SockGetmem`  
    `Ispmac.c`, [190](#)

`Ispmac_SockSendControlCharPrint`  
    `Ispmac.c`, [190](#)

`Ispmac_SockSendDPline`  
    `Ispmac.c`, [190](#)  
    `pgpmac.h`, [284](#)

`Ispmac_SockSendDPqueue`  
    `Ispmac.c`, [191](#)

`Ispmac_SockSendline`  
    `Ispmac.c`, [191](#)  
    `pgpmac.h`, [284](#)

`Ispmac_SockSendline_nr`  
    `Ispmac.c`, [192](#)

`Ispmac_ascii_buffers`  
    `Ispmac.c`, [199](#)

`Ispmac_ascii_buffers_mutex`  
    `Ispmac.c`, [199](#)

`Ispmac_ascii_buffers_struct`, [42](#)  
    `command_buf`, [42](#)  
    `command_buf_cc`, [42](#)  
    `command_str`, [42](#)  
    `response_buf`, [42](#)  
    `response_n`, [42](#)  
    `response_str`, [42](#)

`Ispmac_ascii_buffers_t`  
    `Ispmac.c`, [145](#)

`Ispmac_ascii_busy`  
    `Ispmac.c`, [199](#)

`Ispmac_ascii_mutex`  
    `Ispmac.c`, [199](#)

`Ispmac_asciicmdCB`  
    `Ispmac.c`, [147](#)

`Ispmac_backLight_down_cb`  
    `Ispmac.c`, [147](#)

`Ispmac_backLight_up_cb`  
    `Ispmac.c`, [148](#)

`Ispmac_bi_init`  
    `Ispmac.c`, [148](#)

`Ispmac_bi_struct`, [42](#)  
    `changeEventOff`, [43](#)  
    `changeEventOn`, [43](#)  
    `first_time`, [43](#)  
    `mask`, [43](#)  
    `mutex`, [43](#)  
    `position`, [44](#)  
    `previous`, [44](#)  
    `ptr`, [44](#)

`Ispmac_bi_t`  
    `pgpmac.h`, [264](#)

`Ispmac_bis`  
    `Ispmac.c`, [200](#)

`Ispmac_blight_lut_setup`  
    `Ispmac.c`, [148](#)

`Ispmac_bo_init`  
    `Ispmac.c`, [149](#)

`Ispmac_bo_read`  
    `Ispmac.c`, [149](#)

`Ispmac_cmd_queue_struct`, [44](#)  
    `event`, [45](#)  
    `no_reply`, [45](#)  
    `onResponse`, [45](#)  
    `pcmd`, [45](#)  
    `time_sent`, [45](#)

`Ispmac_command_done_cb`  
    `Ispmac.c`, [150](#)

`Ispmac_cryoSwitchChanged_cb`  
    `Ispmac.c`, [150](#)

`Ispmac_dac_init`  
    `Ispmac.c`, [150](#)

`Ispmac_dac_read`  
    `Ispmac.c`, [151](#)

`Ispmac_dpascii_off`  
    `Ispmac.c`, [200](#)

`Ispmac_dpascii_on`  
    `Ispmac.c`, [200](#)

`Ispmac_dpascii_queue`  
    `Ispmac.c`, [200](#)

`Ispmac_dpascii_queue_struct`, [45](#)  
    `event`, [45](#)  
    `pl`, [45](#)

`Ispmac_dpascii_queue_t`  
    `Ispmac.c`, [145](#)

`Ispmac_flight_lut_setup`  
    `Ispmac.c`, [152](#)

`Ispmac_fscint_lut_setup`  
    `Ispmac.c`, [152](#)

`Ispmac_fshut_init`  
    `Ispmac.c`, [153](#)

`Ispmac_get_ascii`  
    `Ispmac.c`, [153](#)

`Ispmac_get_ascii_cb`  
    `Ispmac.c`, [153](#)

`Ispmac_get_status`  
    `Ispmac.c`, [155](#)

`Ispmac_get_status_cb`  
    `Ispmac.c`, [155](#)

`Ispmac_getBIPosition`  
    `Ispmac.c`, [159](#)  
    `pgpmac.h`, [274](#)

`Ispmac_getPosition`  
    `Ispmac.c`, [160](#)  
    `pgpmac.h`, [274](#)

`Ispmac_home1_queue`  
    `Ispmac.c`, [160](#)  
    `pgpmac.h`, [275](#)

`Ispmac_home2_queue`  
    `Ispmac.c`, [162](#)

- lspmac\_init
  - lspmac.c, 162
  - pgpmac.h, 276
- lspmac\_jogabs\_queue
  - lspmac.c, 165
  - pgpmac.h, 279
- lspmac\_light\_zoom\_cb
  - lspmac.c, 166
- lspmac\_lut
  - lspmac.c, 166
- lspmac\_md2\_init
  - lspmac.c, 200
- lspmac\_more\_ascii\_cb
  - lspmac.c, 167
- lspmac\_motor\_init
  - lspmac.c, 167
- lspmac\_motor\_struct, 46
  - active, 48
  - active\_init, 48
  - actual\_pos\_cnts, 48
  - actual\_pos\_cnts\_p, 48
  - axis, 48
  - command\_sent, 48
  - cond, 48
  - coord\_num, 48
  - dac\_mvar, 48
  - home, 49
  - homing, 49
  - inactive\_init, 49
  - lut, 49
  - max\_accel, 49
  - max\_speed, 49
  - motion\_seen, 49
  - motor\_num, 49
  - moveAbs, 49
  - mutex, 50
  - name, 50
  - nlut, 50
  - not\_done, 50
  - position, 50
  - pq, 50
  - printf\_fmt, 50
  - read, 50
  - read\_mask, 50
  - read\_ptr, 51
  - redis\_fmt, 51
  - redis\_position, 51
  - reported\_position, 51
  - requested\_pos\_cnts, 51
  - requested\_position, 51
  - status1, 51
  - status1\_p, 51
  - status2, 51
  - status2\_p, 52
  - status\_str, 52
  - u2c, 52
  - unit, 52
  - update\_resolution, 52
  - win, 52
  - write\_fmt, 52
- lspmac\_motor\_t
  - pgpmac.h, 264
- lspmac\_motors
  - lspmac.c, 200
  - pgpmac.h, 297
- lspmac\_move\_or\_jog\_abs\_queue
  - lspmac.c, 168
  - pgpmac.h, 279
- lspmac\_move\_or\_jog\_preset\_queue
  - lspmac.c, 170
  - pgpmac.h, 281
- lspmac\_move\_or\_jog\_queue
  - pgpmac.h, 281
- lspmac\_move\_preset\_queue
  - lspmac.c, 170
  - pgpmac.h, 281
- lspmac\_moveabs\_blight\_factor\_queue
  - lspmac.c, 170
- lspmac\_moveabs\_bo\_queue
  - lspmac.c, 171
- lspmac\_moveabs\_flight\_factor\_queue
  - lspmac.c, 171
- lspmac\_moveabs\_frontlight\_oo\_queue
  - lspmac.c, 172
- lspmac\_moveabs\_fshut\_queue
  - lspmac.c, 172
- lspmac\_moveabs\_queue
  - lspmac.c, 172
  - pgpmac.h, 282
- lspmac\_moveabs\_timed\_queue
  - lspmac.c, 173
- lspmac\_moveabs\_wait
  - lspmac.c, 174
  - pgpmac.h, 282
- lspmac\_movedac\_queue
  - lspmac.c, 174
- lspmac\_movezoom\_queue
  - lspmac.c, 175
- lspmac\_moving\_cond
  - lspmac.c, 200
  - pgpmac.h, 297
- lspmac\_moving\_flags
  - lspmac.c, 200
  - pgpmac.h, 298
- lspmac\_moving\_mutex
  - lspmac.c, 200
  - pgpmac.h, 298
- lspmac\_nbis
  - lspmac.c, 200
- lspmac\_next\_state
  - lspmac.c, 175
- lspmac\_nmotors
  - lspmac.c, 201
  - pgpmac.h, 298
- lspmac\_pmacmotor\_read
  - lspmac.c, 177

lspmac\_pop\_queue  
  lspmac.c, [180](#)  
lspmac\_pop\_reply  
  lspmac.c, [180](#)  
lspmac\_push\_queue  
  lspmac.c, [181](#)  
lspmac\_reset\_queue  
  lspmac.c, [181](#)  
lspmac\_rlut  
  lspmac.c, [181](#)  
lspmac\_run  
  lspmac.c, [182](#)  
  pgpmac.h, [282](#)  
lspmac\_sample\_detector\_cb  
  lspg.c, [130](#)  
lspmac\_scint\_dried\_cb  
  lspmac.c, [184](#)  
lspmac\_scint\_inPosition\_cb  
  lspmac.c, [184](#)  
lspmac\_send\_command  
  lspmac.c, [184](#)  
lspmac\_sendcmd  
  lspmac.c, [185](#)  
lspmac\_sendcmd\_nocb  
  lspmac.c, [186](#)  
lspmac\_shutter\_cond  
  lspmac.c, [201](#)  
  pgpmac.h, [298](#)  
lspmac\_shutter\_has\_opened  
  lspmac.c, [201](#)  
  pgpmac.h, [298](#)  
lspmac\_shutter\_mutex  
  lspmac.c, [201](#)  
  pgpmac.h, [298](#)  
lspmac\_shutter\_read  
  lspmac.c, [189](#)  
lspmac\_shutter\_state  
  lspmac.c, [201](#)  
  pgpmac.h, [298](#)  
lspmac\_soft\_motor\_init  
  lspmac.c, [192](#)  
lspmac\_soft\_motor\_read  
  lspmac.c, [192](#)  
lspmac\_status\_last\_time  
  lspmac.c, [201](#)  
lspmac\_status\_time  
  lspmac.c, [201](#)  
lspmac\_test\_preset  
  lspmac.c, [192](#)  
lspmac\_video\_rotate  
  lspmac.c, [193](#)  
  pgpmac.h, [285](#)  
lspmac\_worker  
  lspmac.c, [193](#)  
lspmac\_zoom\_lut\_setup  
  lspmac.c, [194](#)  
lsredis.c, [204](#)  
  \_lsredis\_get\_obj, [207](#)  
  \_lsredis\_set\_value, [208](#)  
lsredis\_addRead, [209](#)  
lsredis\_addWrite, [209](#)  
lsredis\_cleanup, [209](#)  
lsredis\_cmpnstr, [210](#)  
lsredis\_cmpstr, [210](#)  
lsredis\_cond, [221](#)  
lsredis\_debugCB, [210](#)  
lsredis\_delRead, [211](#)  
lsredis\_delWrite, [211](#)  
lsredis\_fd\_service, [211](#)  
lsredis\_find\_preset, [212](#)  
lsredis\_get\_obj, [212](#)  
lsredis\_get\_string\_array, [213](#)  
lsredis\_getb, [213](#)  
lsredis\_getc, [213](#)  
lsredis\_getd, [214](#)  
lsredis\_getl, [214](#)  
lsredis\_getstr, [214](#)  
lsredis\_head, [221](#)  
lsredis\_hgetCB, [214](#)  
lsredis\_hstab, [221](#)  
lsredis\_init, [215](#)  
lsredis\_key\_select\_regex, [222](#)  
lsredis\_keysCB, [216](#)  
lsredis\_maybe\_add\_key, [216](#)  
lsredis\_mutex, [222](#)  
lsredis\_objs, [222](#)  
lsredis\_publisher, [222](#)  
lsredis\_regexec, [217](#)  
lsredis\_run, [217](#)  
lsredis\_running, [222](#)  
lsredis\_set\_value, [217](#)  
lsredis\_setstr, [217](#)  
lsredis\_sig\_service, [218](#)  
lsredis\_subCB, [219](#)  
lsredis\_thread, [222](#)  
lsredis\_worker, [220](#)  
redisDisconnectCB, [221](#)  
roac, [222](#)  
rofd, [222](#)  
subac, [222](#)  
subfd, [222](#)  
wrac, [222](#)  
wrfd, [222](#)  
lsredis\_addRead  
  lsredis.c, [209](#)  
lsredis\_addWrite  
  lsredis.c, [209](#)  
lsredis\_cleanup  
  lsredis.c, [209](#)  
lsredis\_cmpnstr  
  lsredis.c, [210](#)  
  pgpmac.h, [285](#)  
lsredis\_cmpstr  
  lsredis.c, [210](#)  
  pgpmac.h, [285](#)  
lsredis\_cond

- lsredis.c, 221
- lsredis\_debugCB
  - lsredis.c, 210
- lsredis\_delRead
  - lsredis.c, 211
- lsredis\_delWrite
  - lsredis.c, 211
- lsredis\_fd\_service
  - lsredis.c, 211
- lsredis\_find\_preset
  - lsredis.c, 212
  - pgpmac.h, 285
- lsredis\_get\_obj
  - lsredis.c, 212
  - pgpmac.h, 286
- lsredis\_get\_string\_array
  - lsredis.c, 213
  - pgpmac.h, 287
- lsredis\_getb
  - lsredis.c, 213
  - pgpmac.h, 287
- lsredis\_getc
  - lsredis.c, 213
- lsredis\_getd
  - lsredis.c, 214
  - pgpmac.h, 287
- lsredis\_getl
  - lsredis.c, 214
  - pgpmac.h, 287
- lsredis\_getstr
  - lsredis.c, 214
  - pgpmac.h, 288
- lsredis\_head
  - lsredis.c, 221
- lsredis\_hgetCB
  - lsredis.c, 214
- lsredis\_htab
  - lsredis.c, 221
- lsredis\_init
  - lsredis.c, 215
  - pgpmac.h, 288
- lsredis\_key\_select\_regex
  - lsredis.c, 222
- lsredis\_keysCB
  - lsredis.c, 216
- lsredis\_maybe\_add\_key
  - lsredis.c, 216
- lsredis\_mutex
  - lsredis.c, 222
- lsredis\_obj\_struct, 52
  - avalue, 53
  - bvalue, 53
  - cond, 53
  - cvalue, 54
  - dvalue, 54
  - events\_name, 54
  - hits, 54
  - key, 54
  - lvalue, 54
  - mutex, 54
  - next, 54
  - valid, 54
  - value, 55
  - value\_length, 55
  - wait\_for\_me, 55
- lsredis\_obj\_t
  - pgpmac.h, 264
- lsredis\_objs
  - lsredis.c, 222
- lsredis\_publisher
  - lsredis.c, 222
- lsredis\_regex
  - lsredis.c, 217
  - pgpmac.h, 289
- lsredis\_run
  - lsredis.c, 217
  - pgpmac.h, 289
- lsredis\_running
  - lsredis.c, 222
- lsredis\_set\_value
  - lsredis.c, 217
- lsredis\_setstr
  - lsredis.c, 217
  - pgpmac.h, 290
- lsredis\_sig\_service
  - lsredis.c, 218
- lsredis\_subCB
  - lsredis.c, 219
- lsredis\_thread
  - lsredis.c, 222
- lsredis\_worker
  - lsredis.c, 220
- lster.c, 223
  - handler, 224
  - LSTIMER\_LIST\_LENGTH, 224
  - lster\_active\_timers, 228
  - lster\_add\_timer, 225
  - lster\_cond, 228
  - lster\_init, 225
  - lster\_list, 228
  - lster\_list\_t, 224
  - lster\_mutex, 228
  - lster\_run, 226
  - lster\_thread, 228
  - lster\_timerid, 229
  - lster\_worker, 226
  - new\_timer, 229
  - service\_timers, 227
- lster\_active\_timers
  - lster.c, 228
- lster\_add\_timer
  - lster.c, 225
  - pgpmac.h, 290
- lster\_cond
  - lster.c, 228
- lster\_init

- lstimer.c, 225
- pgpmac.h, 291
- lstimer\_list
  - lstimer.c, 228
- lstimer\_list\_struct, 55
  - delay\_nsecs, 56
  - delay\_secs, 56
  - event, 56
  - init\_nsecs, 56
  - init\_secs, 56
  - last\_nsecs, 56
  - last\_secs, 56
  - ncalls, 56
  - next\_nsecs, 57
  - next\_secs, 57
  - shots, 57
- lstimer\_list\_t
  - lstimer.c, 224
- lstimer\_mutex
  - lstimer.c, 228
- lstimer\_run
  - lstimer.c, 226
  - pgpmac.h, 292
- lstimer\_thread
  - lstimer.c, 228
- lstimer\_timerid
  - lstimer.c, 229
- lstimer\_worker
  - lstimer.c, 226
- lupdate\_init
  - pgpmac.h, 292
- lupdate\_run
  - pgpmac.h, 292
- ltime
  - lslogging\_queue\_struct, 19
- lut
  - lspmac\_motor\_struct, 49
- lvalue
  - lsredis\_obj\_struct, 54
- MD2CMDSD\_CMD\_LENGTH
  - pgpmac.h, 263
- main
  - kvredis.c, 79
  - pgpmac.c, 251
- mask
  - lspmac\_bi\_struct, 43
- max\_accel
  - lspmac\_motor\_struct, 49
- max\_speed
  - lspmac\_motor\_struct, 49
- md2\_status
  - lspmac.c, 201
- md2\_status\_mutex
  - lspmac.c, 201
  - pgpmac.h, 298
- md2\_status\_t
  - lspmac.c, 145
- md2StatusStruct, 57
  - acc11c\_1, 59
  - acc11c\_2, 59
  - acc11c\_3, 59
  - acc11c\_5, 59
  - acc11c\_6, 59
  - alignx\_act\_pos, 59
  - alignx\_status\_1, 59
  - alignx\_status\_2, 59
  - aligny\_act\_pos, 59
  - aligny\_status\_1, 59
  - aligny\_status\_2, 59
  - alignz\_act\_pos, 59
  - alignz\_status\_1, 59
  - alignz\_status\_2, 60
  - analyzer\_act\_pos, 60
  - analyzer\_status\_1, 60
  - analyzer\_status\_2, 60
  - aperturey\_act\_pos, 60
  - aperturey\_status\_1, 60
  - aperturey\_status\_2, 60
  - aperturez\_act\_pos, 60
  - aperturez\_status\_1, 60
  - aperturez\_status\_2, 60
  - back\_dac, 60
  - copy\_act\_pos, 60
  - copy\_status\_1, 61
  - copy\_status\_2, 61
  - capz\_act\_pos, 61
  - capz\_status\_1, 61
  - capz\_status\_2, 61
  - centerx\_act\_pos, 61
  - centerx\_status\_1, 61
  - centerx\_status\_2, 61
  - centery\_act\_pos, 61
  - centery\_status\_1, 61
  - centery\_status\_2, 61
  - dummy1, 61
  - dummy2, 62
  - dummy3, 62
  - dummy4, 62
  - dummy5, 62
  - dummy6, 62
  - dummy7, 62
  - dummy8, 62
  - dummy9, 62
  - dummyA, 62
  - dummyB, 62
  - front\_dac, 62
  - fs\_has\_opened, 62
  - fs\_has\_opened\_globally, 63
  - fs\_is\_open, 63
  - kappa\_act\_pos, 63
  - kappa\_status\_1, 63
  - kappa\_status\_2, 63
  - moving\_flags, 63
  - number\_passes, 63
  - omega\_act\_pos, 63
  - omega\_status\_1, 63

- omega\_status\_2, [63](#)
- phi\_act\_pos, [63](#)
- phi\_status\_1, [63](#)
- phi\_status\_2, [64](#)
- phiscan, [64](#)
- scint\_act\_pos, [64](#)
- scint\_piezo, [64](#)
- scint\_status\_1, [64](#)
- scint\_status\_2, [64](#)
- zoom\_act\_pos, [64](#)
- zoom\_status\_1, [64](#)
- zoom\_status\_2, [64](#)
- md2cmds.c, [229](#)
  - md2cmds\_action\_queue, [231](#)
  - md2cmds\_action\_wait, [231](#)
  - md2cmds\_capz\_moving\_time, [248](#)
  - md2cmds\_center, [231](#)
  - md2cmds\_cmd, [248](#)
  - md2cmds\_collect, [231](#)
  - md2cmds\_cond, [249](#)
  - md2cmds\_init, [234](#)
  - md2cmds\_kappaphi\_move, [235](#)
  - md2cmds\_maybe\_done\_moving\_cb, [235](#)
  - md2cmds\_maybe\_rotate\_done\_cb, [235](#)
  - md2cmds\_md\_status\_code, [249](#)
  - md2cmds\_move\_prep, [236](#)
  - md2cmds\_move\_prep\_done\_cb, [237](#)
  - md2cmds\_move\_wait, [237](#)
  - md2cmds\_moveAbs, [238](#)
  - md2cmds\_moving\_cond, [249](#)
  - md2cmds\_moving\_count, [249](#)
  - md2cmds\_moving\_mutex, [249](#)
  - md2cmds\_moving\_pq, [249](#)
  - md2cmds\_moving\_queue\_wait, [249](#)
  - md2cmds\_mutex, [249](#)
  - md2cmds\_mvcenter\_move, [239](#)
  - md2cmds\_organs\_move\_presets, [239](#)
  - md2cmds\_organs\_prep, [240](#)
  - md2cmds\_organs\_wait, [240](#)
  - md2cmds\_phase\_change, [240](#)
  - md2cmds\_prep\_axis, [242](#)
  - md2cmds\_rotate, [243](#)
  - md2cmds\_rotate\_cb, [244](#)
  - md2cmds\_run, [244](#)
  - md2cmds\_set\_scale\_cb, [245](#)
  - md2cmds\_thread, [249](#)
  - md2cmds\_time\_capz\_cb, [245](#)
  - md2cmds\_transfer, [245](#)
  - md2cmds\_worker, [248](#)
  - rotating, [249](#)
- md2cmds\_action\_queue
  - md2cmds.c, [231](#)
- md2cmds\_action\_wait
  - md2cmds.c, [231](#)
- md2cmds\_capz\_moving\_time
  - md2cmds.c, [248](#)
- md2cmds\_center
  - md2cmds.c, [231](#)
- md2cmds\_cmd
  - md2cmds.c, [248](#)
  - pgpmac.h, [298](#)
- md2cmds\_collect
  - md2cmds.c, [231](#)
- md2cmds\_cond
  - md2cmds.c, [249](#)
  - pgpmac.h, [299](#)
- md2cmds\_init
  - md2cmds.c, [234](#)
  - pgpmac.h, [292](#)
- md2cmds\_kappaphi\_move
  - md2cmds.c, [235](#)
- md2cmds\_maybe\_done\_moving\_cb
  - md2cmds.c, [235](#)
- md2cmds\_maybe\_rotate\_done\_cb
  - md2cmds.c, [235](#)
- md2cmds\_md\_status\_code
  - md2cmds.c, [249](#)
  - pgpmac.h, [299](#)
- md2cmds\_move\_prep
  - md2cmds.c, [236](#)
- md2cmds\_move\_prep\_done\_cb
  - md2cmds.c, [237](#)
- md2cmds\_move\_wait
  - md2cmds.c, [237](#)
- md2cmds\_moveAbs
  - md2cmds.c, [238](#)
- md2cmds\_moving\_cond
  - md2cmds.c, [249](#)
- md2cmds\_moving\_count
  - md2cmds.c, [249](#)
- md2cmds\_moving\_mutex
  - md2cmds.c, [249](#)
- md2cmds\_moving\_pq
  - md2cmds.c, [249](#)
- md2cmds\_moving\_queue\_wait
  - md2cmds.c, [249](#)
- md2cmds\_mutex
  - md2cmds.c, [249](#)
  - pgpmac.h, [299](#)
- md2cmds\_mvcenter\_move
  - md2cmds.c, [239](#)
- md2cmds\_organs\_move\_presets
  - md2cmds.c, [239](#)
- md2cmds\_organs\_prep
  - md2cmds.c, [240](#)
- md2cmds\_organs\_wait
  - md2cmds.c, [240](#)
- md2cmds\_pg\_cond
  - pgpmac.h, [299](#)
- md2cmds\_pg\_mutex
  - pgpmac.h, [299](#)
- md2cmds\_phase\_change
  - md2cmds.c, [240](#)
- md2cmds\_prep\_axis
  - md2cmds.c, [242](#)
- md2cmds\_rotate
  - md2cmds.c, [243](#)



- md2cmds.c, [243](#)
- md2cmds\_rotate\_cb
  - md2cmds.c, [244](#)
- md2cmds\_run
  - md2cmds.c, [244](#)
  - pgpmac.h, [292](#)
- md2cmds\_set\_scale\_cb
  - md2cmds.c, [245](#)
- md2cmds\_thread
  - md2cmds.c, [249](#)
- md2cmds\_time\_capz\_cb
  - md2cmds.c, [245](#)
- md2cmds\_transfer
  - md2cmds.c, [245](#)
- md2cmds\_worker
  - md2cmds.c, [248](#)
- minikappa\_ok
  - lspmac.c, [202](#)
  - pgpmac.h, [299](#)
- mk\_pgpmac\_redis, [11](#)
  - b, [12](#)
  - bi\_list, [12](#)
  - f, [12](#)
  - fnc, [12](#)
  - hard\_ini, [12](#)
  - hard\_ini\_fields, [12](#)
  - head, [12](#)
  - hi, [12](#)
  - i, [12](#)
  - motor\_dict, [13](#)
  - motor\_field\_lists, [13](#)
  - motor\_presets, [13](#)
  - p, [13](#)
  - pi, [13](#)
  - ppos, [13](#)
  - pref\_ini, [13](#)
  - v, [13](#)
  - x, [13](#)
  - y, [13](#)
  - zoom\_settings, [13](#)
- mk\_pgpmac\_redis.py, [250](#)
- motion\_seen
  - lspmac\_motor\_struct, [49](#)
- motor\_dict
  - mk\_pgpmac\_redis, [13](#)
- motor\_field\_lists
  - mk\_pgpmac\_redis, [13](#)
- motor\_num
  - lspmac\_motor\_struct, [49](#)
- motor\_presets
  - mk\_pgpmac\_redis, [13](#)
- moveAbs
  - lspmac\_motor\_struct, [49](#)
- moving\_flags
  - md2StatusStruct, [63](#)
- mutex
  - lspg\_demandairrights\_struct, [20](#)
  - lspg\_getcenter\_struct, [22](#)
- lspg\_getcurrentsampleid\_struct, [24](#)
- lspg\_lock\_detector\_struct, [24](#)
- lspg\_lock\_diffractionmeter\_struct, [25](#)
- lspg\_nextsample\_struct, [26](#)
- lspg\_nextshot\_struct, [35](#)
- lspg\_seq\_run\_prep\_struct, [38](#)
- lspg\_starttransfer\_struct, [39](#)
- lspg\_wait\_for\_detector\_struct, [40](#)
- lspg\_waitcryo\_struct, [40](#)
- lspmac\_bi\_struct, [43](#)
- lspmac\_motor\_struct, [50](#)
- lsredis\_obj\_struct, [54](#)
- name
  - lspmac\_motor\_struct, [50](#)
- ncalls
  - lstimer\_list\_struct, [56](#)
- ncurses\_mutex
  - pgpmac.c, [254](#)
  - pgpmac.h, [299](#)
- new\_timer
  - lstimer.c, [229](#)
- new\_value\_ready
  - lspg\_demandairrights\_struct, [20](#)
  - lspg\_getcenter\_struct, [22](#)
  - lspg\_getcurrentsampleid\_struct, [24](#)
  - lspg\_lock\_detector\_struct, [25](#)
  - lspg\_lock\_diffractionmeter\_struct, [25](#)
  - lspg\_nextsample\_struct, [26](#)
  - lspg\_nextshot\_struct, [36](#)
  - lspg\_seq\_run\_prep\_struct, [38](#)
  - lspg\_starttransfer\_struct, [39](#)
  - lspg\_wait\_for\_detector\_struct, [40](#)
  - lspg\_waitcryo\_struct, [41](#)
- next
  - lsevents\_listener\_struct, [18](#)
  - lsredis\_obj\_struct, [54](#)
- next\_nsecs
  - lstimer\_list\_struct, [57](#)
- next\_secs
  - lstimer\_list\_struct, [57](#)
- nextsample
  - lspg\_nextsample\_struct, [26](#)
- nextsample\_isnull
  - lspg\_nextsample\_struct, [26](#)
- nlut
  - lspmac\_motor\_struct, [50](#)
- no\_reply
  - lspmac\_cmd\_queue\_struct, [45](#)
- no\_rows\_returned
  - lspg\_getcenter\_struct, [22](#)
  - lspg\_getcurrentsampleid\_struct, [24](#)
  - lspg\_nextsample\_struct, [26](#)
  - lspg\_nextshot\_struct, [36](#)
  - lspg\_starttransfer\_struct, [39](#)
- not\_done
  - lspmac\_motor\_struct, [50](#)
- now
  - kvredis.c, [82](#)

- lspg.c, [132](#)
- lspmac.c, [202](#)
- number\_passes
  - md2StatusStruct, [63](#)
- omega
  - lspmac.c, [202](#)
  - pgpmac.h, [299](#)
- omega\_act\_pos
  - md2StatusStruct, [63](#)
- omega\_status\_1
  - md2StatusStruct, [63](#)
- omega\_status\_2
  - md2StatusStruct, [63](#)
- omega\_zero\_search
  - lspmac.c, [202](#)
- omega\_zero\_time
  - lspmac.c, [202](#)
  - pgpmac.h, [299](#)
- omega\_zero\_velocity
  - lspmac.c, [202](#)
- onResponse
  - lspgQueryQueueStruct, [41](#)
  - lspmac\_cmd\_queue\_struct, [45](#)
- options
  - iniParser::iniParser, [16](#)
- p
  - mk\_pgpmac\_redis, [13](#)
- PMAC\_MIN\_CMD\_TIME
  - lspmac.c, [143](#)
- PMACPORT
  - lspmac.c, [143](#)
- pcmd
  - lspmac\_cmd\_queue\_struct, [45](#)
- pgpmac.c, [250](#)
  - main, [251](#)
  - ncurses\_mutex, [254](#)
  - pgpmac\_printf, [253](#)
  - running, [254](#)
  - stdinService, [253](#)
  - stdinfda, [255](#)
  - term\_input, [255](#)
  - term\_output, [255](#)
  - term\_status, [255](#)
  - term\_status2, [255](#)
- pgpmac.h, [255](#)
  - \_GNU\_SOURCE, [263](#)
  - alignx, [293](#)
  - aligny, [293](#)
  - alignz, [293](#)
  - anal, [293](#)
  - apery, [293](#)
  - aperz, [293](#)
  - arm\_parked, [294](#)
  - blight, [294](#)
  - blight\_down, [294](#)
  - blight\_f, [294](#)
  - blight\_ud, [294](#)
  - blight\_up, [294](#)
  - capy, [294](#)
  - capz, [294](#)
  - cenx, [294](#)
  - ceny, [295](#)
  - cryo, [295](#)
  - cryo\_back, [295](#)
  - cryo\_switch, [295](#)
  - dryer, [295](#)
  - etel\_init\_ok, [295](#)
  - etel\_on, [295](#)
  - etel\_ready, [295](#)
  - flight, [295](#)
  - flight\_f, [296](#)
  - flight\_oo, [296](#)
  - fluo, [296](#)
  - fluor\_back, [296](#)
  - fscint, [296](#)
  - fshut, [296](#)
  - hp\_air, [296](#)
  - kappa, [296](#)
  - lp\_air, [296](#)
  - lsevents\_add\_listener, [265](#)
  - lsevents\_init, [265](#)
  - lsevents\_remove\_listener, [266](#)
  - lsevents\_run, [266](#)
  - lsevents\_send\_event, [266](#)
  - lslogging\_init, [267](#)
  - lslogging\_log\_message, [267](#)
  - lslogging\_run, [268](#)
  - lspg\_array2ptrs, [268](#)
  - lspg\_demandairrights, [297](#)
  - lspg\_demandairrights\_all, [269](#)
  - lspg\_demandairrights\_t, [264](#)
  - lspg\_getcenter, [297](#)
  - lspg\_getcenter\_call, [270](#)
  - lspg\_getcenter\_done, [270](#)
  - lspg\_getcenter\_t, [264](#)
  - lspg\_getcenter\_wait, [270](#)
  - lspg\_getcurrentsampleid, [297](#)
  - lspg\_getcurrentsampleid\_t, [264](#)
  - lspg\_getcurrentsampleid\_wait\_for\_id, [270](#)
  - lspg\_init, [270](#)
  - lspg\_nextsample, [297](#)
  - lspg\_nextsample\_all, [271](#)
  - lspg\_nextsample\_t, [264](#)
  - lspg\_nextshot, [297](#)
  - lspg\_nextshot\_call, [271](#)
  - lspg\_nextshot\_done, [271](#)
  - lspg\_nextshot\_t, [264](#)
  - lspg\_nextshot\_wait, [272](#)
  - lspg\_query\_push, [272](#)
  - lspg\_query\_queue\_t, [264](#)
  - lspg\_run, [272](#)
  - lspg\_seq\_run\_prep\_all, [273](#)
  - lspg\_starttransfer, [297](#)
  - lspg\_starttransfer\_call, [273](#)
  - lspg\_starttransfer\_done, [273](#)

- lspg\_starttransfer\_t, 264
- lspg\_starttransfer\_wait, 273
- lspg\_waitcryo, 297
- lspg\_waitcryo\_all, 274
- lspg\_waitcryo\_cb, 274
- lspg\_waitcryo\_t, 264
- lspg\_zoom\_lut\_call, 274
- lspmac\_SockSendDPLine, 284
- lspmac\_SockSendline, 284
- lspmac\_bi\_t, 264
- lspmac\_getBIPosition, 274
- lspmac\_getPosition, 274
- lspmac\_home1\_queue, 275
- lspmac\_init, 276
- lspmac\_jogabs\_queue, 279
- lspmac\_motor\_t, 264
- lspmac\_motors, 297
- lspmac\_move\_or\_jog\_abs\_queue, 279
- lspmac\_move\_or\_jog\_preset\_queue, 281
- lspmac\_move\_or\_jog\_queue, 281
- lspmac\_move\_preset\_queue, 281
- lspmac\_moveabs\_queue, 282
- lspmac\_moveabs\_wait, 282
- lspmac\_moving\_cond, 297
- lspmac\_moving\_flags, 298
- lspmac\_moving\_mutex, 298
- lspmac\_nmotors, 298
- lspmac\_run, 282
- lspmac\_shutter\_cond, 298
- lspmac\_shutter\_has\_opened, 298
- lspmac\_shutter\_mutex, 298
- lspmac\_shutter\_state, 298
- lspmac\_video\_rotate, 285
- lsredis\_cmpnstr, 285
- lsredis\_cmpstr, 285
- lsredis\_find\_preset, 285
- lsredis\_get\_obj, 286
- lsredis\_get\_string\_array, 287
- lsredis\_getb, 287
- lsredis\_getd, 287
- lsredis\_getl, 287
- lsredis\_getstr, 288
- lsredis\_init, 288
- lsredis\_obj\_t, 264
- lsredis\_regexec, 289
- lsredis\_run, 289
- lsredis\_setstr, 290
- lster\_add\_timer, 290
- lster\_init, 291
- lster\_run, 292
- lsupdate\_init, 292
- lsupdate\_run, 292
- MD2CMDS\_CMD\_LENGTH, 263
- md2\_status\_mutex, 298
- md2cmds\_cmd, 298
- md2cmds\_cond, 299
- md2cmds\_init, 292
- md2cmds\_md\_status\_code, 299
- md2cmds\_mutex, 299
- md2cmds\_pg\_cond, 299
- md2cmds\_pg\_mutex, 299
- md2cmds\_run, 292
- minikappa\_ok, 299
- ncurses\_mutex, 299
- omega, 299
- omega\_zero\_time, 299
- pgpmac\_printf, 292
- phi, 299
- pmac\_cmd\_queue\_t, 264
- pmac\_cmd\_t, 264
- pmac\_queue\_cond, 299
- pmac\_queue\_mutex, 300
- PmacSockSendline, 293
- sample\_detected, 300
- scint, 300
- shutter\_open, 300
- smart\_mag\_err, 300
- smart\_mag\_off, 300
- smart\_mag\_on, 300
- smart\_mag\_oo, 300
- term\_input, 300
- term\_output, 301
- term\_status, 301
- term\_status2, 301
- zoom, 301
- pgpmac\_printf
  - pgpmac.c, 253
  - pgpmac.h, 292
- phi
  - lspmac.c, 202
  - pgpmac.h, 299
- phi\_act\_pos
  - md2StatusStruct, 63
- phi\_status\_1
  - md2StatusStruct, 63
- phi\_status\_2
  - md2StatusStruct, 64
- phiscan
  - md2StatusStruct, 64
- pi
  - mk\_pgpmac\_redis, 13
- pl
  - lspmac\_dpascii\_queue\_struct, 45
- pmac\_cmd\_queue\_t
  - pgpmac.h, 264
- pmac\_cmd\_size
  - lspmac.c, 143
- pmac\_cmd\_t
  - pgpmac.h, 264
- pmac\_error\_strs
  - lspmac.c, 202
- pmac\_queue\_cond
  - lspmac.c, 203
  - pgpmac.h, 299
- pmac\_queue\_mutex
  - lspmac.c, 203

- pgpmac.h, 300
- pmac\_thread
  - lspmac.c, 203
- PmacSockSendline
  - pgpmac.h, 293
- pmacfd
  - lspmac.c, 203
- position
  - lspmac\_bi\_struct, 44
  - lspmac\_motor\_struct, 50
- ppos
  - mk\_pgpmac\_redis, 13
- pq
  - lspmac\_motor\_struct, 50
- pref\_ini
  - mk\_pgpmac\_redis, 13
- previous
  - lspmac\_bi\_struct, 44
- printf\_fmt
  - lspmac\_motor\_struct, 50
- ptr
  - lspmac\_bi\_struct, 44
- q
  - kvredis.c, 82
  - lspg.c, 132
- qs
  - lspgQueryQueueStruct, 41
- raw\_regexp
  - lsevents\_listener\_struct, 18
- re
  - lsevents\_listener\_struct, 18
- read
  - iniParser::iniParser, 16
  - lspmac\_motor\_struct, 50
- read\_mask
  - lspmac\_motor\_struct, 50
- read\_ptr
  - lspmac\_motor\_struct, 51
- redis\_fmt
  - lspmac\_motor\_struct, 51
- redis\_position
  - lspmac\_motor\_struct, 51
- redisDisconnectCB
  - kvredis.c, 81
  - lsredis.c, 221
- reported\_position
  - lspmac\_motor\_struct, 51
- Request
  - tagEthernetCmd, 65
- RequestType
  - tagEthernetCmd, 65
- requested\_pos\_cnts
  - lspmac\_motor\_struct, 51
- requested\_position
  - lspmac\_motor\_struct, 51
- response\_buf
  - lspmac\_ascii\_buffers\_struct, 42
- response\_n
  - lspmac\_ascii\_buffers\_struct, 42
- response\_str
  - lspmac\_ascii\_buffers\_struct, 42
- roac
  - lsredis.c, 222
- rofd
  - lsredis.c, 222
- rotating
  - md2cmds.c, 249
- rr\_cmd
  - lspmac.c, 203
- running
  - pgpmac.c, 254
- sample\_detected
  - lspmac.c, 203
  - pgpmac.h, 300
- scint
  - lspmac.c, 203
  - pgpmac.h, 300
- scint\_act\_pos
  - md2StatusStruct, 64
- scint\_piezo
  - md2StatusStruct, 64
- scint\_status\_1
  - md2StatusStruct, 64
- scint\_status\_2
  - md2StatusStruct, 64
- sd
  - iniParser::iniParser, 17
- sections
  - iniParser::iniParser, 17
- service\_timers
  - lstimer.c, 227
- sfn
  - lspg\_nextshot\_struct, 36
- sfn\_isnull
  - lspg\_nextshot\_struct, 36
- shots
  - lstimer\_list\_struct, 57
- shutter\_open
  - lspmac.c, 204
  - pgpmac.h, 300
- sindex
  - lspg\_nextshot\_struct, 36
- sindex2
  - lspg\_nextshot\_struct, 36
- sindex2\_isnull
  - lspg\_nextshot\_struct, 36
- sindex\_isnull
  - lspg\_nextshot\_struct, 36
- skey
  - lspg\_nextshot\_struct, 36
- skey\_isnull
  - lspg\_nextshot\_struct, 36
- smart\_mag\_err
  - lspmac.c, 204
  - pgpmac.h, 300

- smart\_mag\_off
  - lspmac.c, [204](#)
  - pgpmac.h, [300](#)
- smart\_mag\_on
  - lspmac.c, [204](#)
  - pgpmac.h, [300](#)
- smart\_mag\_oo
  - lspmac.c, [204](#)
  - pgpmac.h, [300](#)
- sstart
  - lspg\_nextshot\_struct, [37](#)
- sstart2
  - lspg\_nextshot\_struct, [37](#)
- sstart2\_isnull
  - lspg\_nextshot\_struct, [37](#)
- sstart\_isnull
  - lspg\_nextshot\_struct, [37](#)
- starttransfer
  - lspg\_starttransfer\_struct, [39](#)
- status1
  - lspmac\_motor\_struct, [51](#)
- status1\_p
  - lspmac\_motor\_struct, [51](#)
- status2
  - lspmac\_motor\_struct, [51](#)
- status2\_p
  - lspmac\_motor\_struct, [52](#)
- status\_str
  - lspmac\_motor\_struct, [52](#)
- stdinService
  - pgpmac.c, [253](#)
- stdinfda
  - pgpmac.c, [255](#)
- stype
  - lspg\_nextshot\_struct, [37](#)
- stype2
  - lspg\_nextshot\_struct, [37](#)
- stype2\_isnull
  - lspg\_nextshot\_struct, [37](#)
- stype\_isnull
  - lspg\_nextshot\_struct, [37](#)
- subac
  - kvredis.c, [83](#)
  - lsredis.c, [222](#)
- subfd
  - kvredis.c, [83](#)
  - lsredis.c, [222](#)
- tagEthernetCmd, [64](#)
  - bData, [65](#)
  - Request, [65](#)
  - RequestType, [65](#)
  - wIndex, [65](#)
  - wLength, [65](#)
  - wValue, [65](#)
- term\_input
  - pgpmac.c, [255](#)
  - pgpmac.h, [300](#)
- term\_output
  - pgpmac.c, [255](#)
  - pgpmac.h, [301](#)
- term\_status
  - pgpmac.c, [255](#)
  - pgpmac.h, [301](#)
- term\_status2
  - pgpmac.c, [255](#)
  - pgpmac.h, [301](#)
- time\_sent
  - lspmac\_cmd\_queue\_struct, [45](#)
- u2c
  - lspmac\_motor\_struct, [52](#)
- unit
  - lspmac\_motor\_struct, [52](#)
- update\_resolution
  - lspmac\_motor\_struct, [52](#)
- v
  - mk\_pgpmac\_redis, [13](#)
- VR\_CTRL\_RESPONSE
  - lspmac.c, [143](#)
- VR\_DOWNLOAD
  - lspmac.c, [143](#)
- VR\_FWDOWNLOAD
  - lspmac.c, [143](#)
- VR\_IPADDRESS
  - lspmac.c, [143](#)
- VR\_PMAC\_FLUSH
  - lspmac.c, [143](#)
- VR\_PMAC\_GETBUFFER
  - lspmac.c, [143](#)
- VR\_PMAC\_GETLINE
  - lspmac.c, [143](#)
- VR\_PMAC\_GETMEM
  - lspmac.c, [144](#)
- VR\_PMAC\_GETRESPONSE
  - lspmac.c, [144](#)
- VR\_PMAC\_PORT
  - lspmac.c, [144](#)
- VR\_PMAC\_READREADY
  - lspmac.c, [144](#)
- VR\_PMAC\_SENDCTRLCHAR
  - lspmac.c, [144](#)
- VR\_PMAC\_SENDLINE
  - lspmac.c, [144](#)
- VR\_PMAC\_SETBIT
  - lspmac.c, [144](#)
- VR\_PMAC\_SETBITS
  - lspmac.c, [144](#)
- VR\_PMAC\_SETMEM
  - lspmac.c, [144](#)
- VR\_PMAC\_WRITEBUFFER
  - lspmac.c, [144](#)
- VR\_PMAC\_WRITEERROR
  - lspmac.c, [144](#)
- VR\_UPLOAD
  - lspmac.c, [144](#)
- valid

- lsredis\_obj\_struct, [54](#)
- value
  - lsredis\_obj\_struct, [55](#)
- value\_length
  - lsredis\_obj\_struct, [55](#)
- wIndex
  - tagEthernetCmd, [65](#)
- wLength
  - tagEthernetCmd, [65](#)
- wValue
  - tagEthernetCmd, [65](#)
- wait\_for\_me
  - lsredis\_obj\_struct, [55](#)
- win
  - lspmac\_motor\_struct, [52](#)
- wrac
  - lsredis.c, [222](#)
- wrfd
  - lsredis.c, [222](#)
- write\_fmt
  - lspmac\_motor\_struct, [52](#)
- x
  - mk\_pgpmac\_redis, [13](#)
- y
  - mk\_pgpmac\_redis, [13](#)
- zoom
  - lspg\_getcenter\_struct, [23](#)
  - lspmac.c, [204](#)
  - pgpmac.h, [301](#)
- zoom\_act\_pos
  - md2StatusStruct, [64](#)
- zoom\_isnull
  - lspg\_getcenter\_struct, [23](#)
- zoom\_settings
  - mk\_pgpmac\_redis, [13](#)
- zoom\_status\_1
  - md2StatusStruct, [64](#)
- zoom\_status\_2
  - md2StatusStruct, [64](#)