

LS-CAT PGPMAC

Generated by Doxygen 1.8.2

Fri Dec 14 2012 16:43:05

Contents

1	The LS-CAT pgpmac Project	1
2	Data Structure Index	5
2.1	Data Structures	5
3	File Index	7
3.1	File List	7
4	Data Structure Documentation	9
4.1	Isevents_listener_struct Struct Reference	9
4.1.1	Detailed Description	9
4.1.2	Field Documentation	9
4.1.2.1	cb	9
4.1.2.2	next	9
4.1.2.3	raw_regexp	10
4.1.2.4	re	10
4.2	Isevents_queue_struct Struct Reference	10
4.2.1	Detailed Description	10
4.2.2	Field Documentation	10
4.2.2.1	event	10
4.3	Iskvs_kvs_list_struct Struct Reference	10
4.3.1	Detailed Description	11
4.3.2	Field Documentation	11
4.3.2.1	kvs	11
4.3.2.2	next	11
4.4	Iskvs_kvs_struct Struct Reference	11
4.4.1	Detailed Description	12
4.4.2	Field Documentation	12
4.4.2.1	k	12
4.4.2.2	l	12
4.4.2.3	next	12
4.4.2.4	v	12

4.4.2.5	vl	12
4.5	Islogging_queue_struct Struct Reference	12
4.5.1	Detailed Description	13
4.5.2	Field Documentation	13
4.5.2.1	lmsg	13
4.5.2.2	ltime	13
4.6	lspg_getcenter_struct Struct Reference	13
4.6.1	Detailed Description	14
4.6.2	Field Documentation	14
4.6.2.1	cond	14
4.6.2.2	dax	14
4.6.2.3	dax_isnull	14
4.6.2.4	day	14
4.6.2.5	day_isnull	14
4.6.2.6	daz	14
4.6.2.7	daz_isnull	14
4.6.2.8	dcx	15
4.6.2.9	dcx_isnull	15
4.6.2.10	dcy	15
4.6.2.11	dcy_isnull	15
4.6.2.12	mutex	15
4.6.2.13	new_value_ready	15
4.6.2.14	no_rows_returned	15
4.6.2.15	zoom	15
4.6.2.16	zoom_isnull	15
4.7	lspg_lock_detector_struct Struct Reference	15
4.7.1	Detailed Description	16
4.7.2	Field Documentation	16
4.7.2.1	cond	16
4.7.2.2	mutex	16
4.7.2.3	new_value_ready	16
4.8	lspg_lock_diffractionmeter_struct Struct Reference	16
4.8.1	Detailed Description	16
4.8.2	Field Documentation	16
4.8.2.1	cond	16
4.8.2.2	mutex	17
4.8.2.3	new_value_ready	17
4.9	lspg_nextshot_struct Struct Reference	17
4.9.1	Detailed Description	19
4.9.2	Field Documentation	20

4.9.2.1	active	20
4.9.2.2	active2	20
4.9.2.3	active2_isnull	20
4.9.2.4	active_isnull	20
4.9.2.5	ax	20
4.9.2.6	ax2	20
4.9.2.7	ax2_isnull	20
4.9.2.8	ax_isnull	20
4.9.2.9	ay	20
4.9.2.10	ay2	20
4.9.2.11	ay2_isnull	21
4.9.2.12	ay_isnull	21
4.9.2.13	az	21
4.9.2.14	az2	21
4.9.2.15	az2_isnull	21
4.9.2.16	az_isnull	21
4.9.2.17	cond	21
4.9.2.18	cx	21
4.9.2.19	cx2	21
4.9.2.20	cx2_isnull	21
4.9.2.21	cx_isnull	21
4.9.2.22	cy	22
4.9.2.23	cy2	22
4.9.2.24	cy2_isnull	22
4.9.2.25	cy_isnull	22
4.9.2.26	dsdir	22
4.9.2.27	dsdir_isnull	22
4.9.2.28	dsdist	22
4.9.2.29	dsdist2	22
4.9.2.30	dsdist2_isnull	22
4.9.2.31	dsdist_isnull	22
4.9.2.32	dsexp	23
4.9.2.33	dsexp2	23
4.9.2.34	dsexp2_isnull	23
4.9.2.35	dsexp_isnull	23
4.9.2.36	dshpid	23
4.9.2.37	dshpid_isnull	23
4.9.2.38	dskappa	23
4.9.2.39	dskappa2	23
4.9.2.40	dskappa2_isnull	23

4.9.2.41	dskappa_isnull	23
4.9.2.42	dsnrg	24
4.9.2.43	dsnrg2	24
4.9.2.44	dsnrg2_isnull	24
4.9.2.45	dsnrg_isnull	24
4.9.2.46	dsomega	24
4.9.2.47	dsomega2	24
4.9.2.48	dsomega2_isnull	24
4.9.2.49	dsomega_isnull	24
4.9.2.50	dsoscaxis	24
4.9.2.51	dsoscaxis2	24
4.9.2.52	dsoscaxis2_isnull	25
4.9.2.53	dsoscaxis_isnull	25
4.9.2.54	dsowidth	25
4.9.2.55	dsowidth2	25
4.9.2.56	dsowidth2_isnull	25
4.9.2.57	dsowidth_isnull	25
4.9.2.58	dsphi	25
4.9.2.59	dsphi2	25
4.9.2.60	dsphi2_isnull	25
4.9.2.61	dsphi_isnull	25
4.9.2.62	dspid	25
4.9.2.63	dspid_isnull	26
4.9.2.64	mutex	26
4.9.2.65	new_value_ready	26
4.9.2.66	no_rows_returned	26
4.9.2.67	sfn	26
4.9.2.68	sfn_isnull	26
4.9.2.69	sindex	26
4.9.2.70	sindex2	26
4.9.2.71	sindex2_isnull	26
4.9.2.72	sindex_isnull	26
4.9.2.73	skey	27
4.9.2.74	skey_isnull	27
4.9.2.75	sstart	27
4.9.2.76	sstart2	27
4.9.2.77	sstart2_isnull	27
4.9.2.78	sstart_isnull	27
4.9.2.79	stype	27
4.9.2.80	stype2	27

4.9.2.81	stype2_isnull	27
4.9.2.82	stype_isnull	27
4.10	lspg_seq_run_prep_struct Struct Reference	28
4.10.1	Detailed Description	28
4.10.2	Field Documentation	28
4.10.2.1	cond	28
4.10.2.2	mutex	28
4.10.2.3	new_value_ready	28
4.11	lspg_wait_for_detector_struct Struct Reference	28
4.11.1	Detailed Description	28
4.11.2	Field Documentation	29
4.11.2.1	cond	29
4.11.2.2	mutex	29
4.11.2.3	new_value_ready	29
4.12	lspgQueryQueueStruct Struct Reference	29
4.12.1	Detailed Description	29
4.12.2	Field Documentation	29
4.12.2.1	onResponse	29
4.12.2.2	qs	29
4.13	lspmac_bi_struct Struct Reference	30
4.13.1	Detailed Description	30
4.13.2	Field Documentation	30
4.13.2.1	changeEventOff	30
4.13.2.2	changeEventOn	30
4.13.2.3	first_time	30
4.13.2.4	mask	31
4.13.2.5	mutex	31
4.13.2.6	previous	31
4.13.2.7	ptr	31
4.14	lspmac_cmd_queue_struct Struct Reference	31
4.14.1	Detailed Description	31
4.14.2	Field Documentation	32
4.14.2.1	no_reply	32
4.14.2.2	onResponse	32
4.14.2.3	pcmd	32
4.14.2.4	rbuff	32
4.14.2.5	time_sent	32
4.15	lspmac_motor_struct Struct Reference	32
4.15.1	Detailed Description	34
4.15.2	Field Documentation	34

4.15.2.1	actual_pos_cnts	34
4.15.2.2	actual_pos_cnts_p	34
4.15.2.3	axis	34
4.15.2.4	cond	34
4.15.2.5	coord_num	35
4.15.2.6	dac_mvar	35
4.15.2.7	format	35
4.15.2.8	home	35
4.15.2.9	homing	35
4.15.2.10	lspg_initialized	35
4.15.2.11	lut	35
4.15.2.12	max_accel	35
4.15.2.13	max_speed	35
4.15.2.14	motion_seen	36
4.15.2.15	motor_num	36
4.15.2.16	moveAbs	36
4.15.2.17	mutex	36
4.15.2.18	name	36
4.15.2.19	nlut	36
4.15.2.20	not_done	36
4.15.2.21	position	36
4.15.2.22	pq	36
4.15.2.23	preset_regex	37
4.15.2.24	presets	37
4.15.2.25	read	37
4.15.2.26	read_mask	37
4.15.2.27	read_ptr	37
4.15.2.28	reported_position	37
4.15.2.29	requested_pos_cnts	37
4.15.2.30	requested_position	37
4.15.2.31	status1	37
4.15.2.32	status1_p	38
4.15.2.33	status2	38
4.15.2.34	status2_p	38
4.15.2.35	statuss	38
4.15.2.36	u2c	38
4.15.2.37	units	38
4.15.2.38	update_format	38
4.15.2.39	update_resolution	38
4.15.2.40	win	38

4.15.2.41	write_fmt	39
4.16	Isredis_obj_struct Struct Reference	39
4.16.1	Detailed Description	39
4.16.2	Field Documentation	40
4.16.2.1	cond	40
4.16.2.2	events_name	40
4.16.2.3	getd	40
4.16.2.4	getl	40
4.16.2.5	getstr	40
4.16.2.6	hits	40
4.16.2.7	key	40
4.16.2.8	mutex	40
4.16.2.9	next	40
4.16.2.10	valid	41
4.16.2.11	value	41
4.16.2.12	value_length	41
4.16.2.13	wait_for_me	41
4.17	Istimer_list_struct Struct Reference	41
4.17.1	Detailed Description	42
4.17.2	Field Documentation	42
4.17.2.1	delay_nsecs	42
4.17.2.2	delay_secs	42
4.17.2.3	event	42
4.17.2.4	init_nsecs	42
4.17.2.5	init_secs	42
4.17.2.6	last_nsecs	42
4.17.2.7	last_secs	42
4.17.2.8	ncalls	43
4.17.2.9	next_nsecs	43
4.17.2.10	next_secs	43
4.17.2.11	shots	43
4.18	md2StatusStruct Struct Reference	43
4.18.1	Detailed Description	44
4.18.2	Field Documentation	45
4.18.2.1	acc11c_1	45
4.18.2.2	acc11c_2	45
4.18.2.3	acc11c_3	45
4.18.2.4	acc11c_5	45
4.18.2.5	acc11c_6	45
4.18.2.6	alignx_act_pos	45

4.18.2.7 alignx_status_1	45
4.18.2.8 alignx_status_2	45
4.18.2.9 aligny_act_pos	45
4.18.2.10 aligny_status_1	45
4.18.2.11 aligny_status_2	45
4.18.2.12 alignz_act_pos	45
4.18.2.13 alignz_status_1	46
4.18.2.14 alignz_status_2	46
4.18.2.15 analyzer_act_pos	46
4.18.2.16 analyzer_status_1	46
4.18.2.17 analyzer_status_2	46
4.18.2.18 aperturey_act_pos	46
4.18.2.19 aperturey_status_1	46
4.18.2.20 aperturey_status_2	46
4.18.2.21 aperturez_act_pos	46
4.18.2.22 aperturez_status_1	46
4.18.2.23 aperturez_status_2	46
4.18.2.24 back_dac	46
4.18.2.25 capy_act_pos	47
4.18.2.26 capy_status_1	47
4.18.2.27 capy_status_2	47
4.18.2.28 capz_act_pos	47
4.18.2.29 capz_status_1	47
4.18.2.30 capz_status_2	47
4.18.2.31 centerx_act_pos	47
4.18.2.32 centerx_status_1	47
4.18.2.33 centerx_status_2	47
4.18.2.34 centery_act_pos	47
4.18.2.35 centery_status_1	47
4.18.2.36 centery_status_2	47
4.18.2.37 dummy1	48
4.18.2.38 dummy2	48
4.18.2.39 dummy3	48
4.18.2.40 dummy4	48
4.18.2.41 dummy5	48
4.18.2.42 dummy6	48
4.18.2.43 dummy7	48
4.18.2.44 dummy8	48
4.18.2.45 dummy9	48
4.18.2.46 dummyA	48

4.18.2.47 dummyB	48
4.18.2.48 front_dac	48
4.18.2.49 fs_has_opened	49
4.18.2.50 fs_has_opened_globally	49
4.18.2.51 fs_is_open	49
4.18.2.52 kappa_act_pos	49
4.18.2.53 kappa_status_1	49
4.18.2.54 kappa_status_2	49
4.18.2.55 moving_flags	49
4.18.2.56 number_passes	49
4.18.2.57 omega_act_pos	49
4.18.2.58 omega_status_1	49
4.18.2.59 omega_status_2	49
4.18.2.60 phi_act_pos	49
4.18.2.61 phi_status_1	50
4.18.2.62 phi_status_2	50
4.18.2.63 phiscan	50
4.18.2.64 scint_act_pos	50
4.18.2.65 scint_piezo	50
4.18.2.66 scint_status_1	50
4.18.2.67 scint_status_2	50
4.18.2.68 zoom_act_pos	50
4.18.2.69 zoom_status_1	50
4.18.2.70 zoom_status_2	50
4.19 tagEthernetCmd Struct Reference	50
4.19.1 Detailed Description	51
4.19.2 Field Documentation	51
4.19.2.1 bData	51
4.19.2.2 Request	51
4.19.2.3 RequestType	51
4.19.2.4 wIndex	51
4.19.2.5 wLength	51
4.19.2.6 wValue	52
5 File Documentation	53
5.1 kvredis.c File Reference	53
5.1.1 Macro Definition Documentation	55
5.1.1.1 LS_PG_QUERY_QUEUE_LENGTH	55
5.1.1.2 LS_PG_QUERY_STRING_LENGTH	55
5.1.1.3 LS_PG_STATE_IDLE	55

5.1.1.4	LS_PG_STATE_INIT	55
5.1.1.5	LS_PG_STATE_INIT_POLL	55
5.1.1.6	LS_PG_STATE_RECV	55
5.1.1.7	LS_PG_STATE_RESET	55
5.1.1.8	LS_PG_STATE_RESET_POLL	55
5.1.1.9	LS_PG_STATE_SEND	56
5.1.1.10	LS_PG_STATE_SEND_FLUSH	56
5.1.2	Typedef Documentation	56
5.1.2.1	lspg_query_queue_t	56
5.1.3	Function Documentation	56
5.1.3.1	addRead	56
5.1.3.2	addWrite	56
5.1.3.3	cleanup	56
5.1.3.4	debugCB	56
5.1.3.5	delRead	57
5.1.3.6	delWrite	57
5.1.3.7	fd_service	57
5.1.3.8	lspg_allkvs_cb	58
5.1.3.9	lspg_flush	58
5.1.3.10	lspg_next_state	59
5.1.3.11	lspg_notice_processor	60
5.1.3.12	lspg_pg_connect	60
5.1.3.13	lspg_pg_service	61
5.1.3.14	lspg_query_next	62
5.1.3.15	lspg_query_push	63
5.1.3.16	lspg_query_reply_next	63
5.1.3.17	lspg_query_reply_peek	63
5.1.3.18	lspg_receive	64
5.1.3.19	lspg_send_next_query	64
5.1.3.20	main	65
5.1.3.21	redisDisconnectCB	67
5.1.4	Variable Documentation	67
5.1.4.1	cmdac	67
5.1.4.2	cmdfd	67
5.1.4.3	kvseq	67
5.1.4.4	ls_pg_state	67
5.1.4.5	lspg_connectPoll_response	67
5.1.4.6	lspg_query_queue	67
5.1.4.7	lspg_query_queue_off	68
5.1.4.8	lspg_query_queue_on	68

5.1.4.9	<code>lspg_query_queue_reply</code>	68
5.1.4.10	<code>lspg_resetPoll_response</code>	68
5.1.4.11	<code>lspgfd</code>	68
5.1.4.12	<code>now</code>	68
5.1.4.13	<code>q</code>	68
5.1.4.14	<code>subac</code>	68
5.1.4.15	<code>subfd</code>	68
5.2	<code>lsevents.c</code> File Reference	69
5.2.1	Detailed Description	70
5.2.2	Macro Definition Documentation	70
5.2.2.1	<code>LSEVENTS_QUEUE_LENGTH</code>	70
5.2.3	Typedef Documentation	70
5.2.3.1	<code>lsevents_listener_t</code>	70
5.2.3.2	<code>lsevents_queue_t</code>	70
5.2.4	Function Documentation	70
5.2.4.1	<code>lsevents_add_listener</code>	70
5.2.4.2	<code>lsevents_init</code>	71
5.2.4.3	<code>lsevents_remove_listener</code>	71
5.2.4.4	<code>lsevents_run</code>	72
5.2.4.5	<code>lsevents_send_event</code>	72
5.2.4.6	<code>lsevents_worker</code>	73
5.2.5	Variable Documentation	73
5.2.5.1	<code>lsevents_listener_mutex</code>	74
5.2.5.2	<code>lsevents_listeners_p</code>	74
5.2.5.3	<code>lsevents_queue</code>	74
5.2.5.4	<code>lsevents_queue_cond</code>	74
5.2.5.5	<code>lsevents_queue_mutex</code>	74
5.2.5.6	<code>lsevents_queue_off</code>	74
5.2.5.7	<code>lsevents_queue_on</code>	74
5.2.5.8	<code>lsevents_thread</code>	74
5.3	<code>lskvs.c</code> File Reference	74
5.3.1	Detailed Description	75
5.3.2	Function Documentation	75
5.3.2.1	<code>lskvs_find_preset_position</code>	75
5.3.2.2	<code>lskvs_get</code>	76
5.3.2.3	<code>lskvs_init</code>	77
5.3.2.4	<code>lskvs_regcomp</code>	77
5.3.2.5	<code>lskvs_run</code>	78
5.3.2.6	<code>lskvs_set</code>	78
5.3.3	Variable Documentation	80

5.3.3.1	lskvs_kvs	80
5.3.3.2	lskvs_rwlock	80
5.4	lslogging.c File Reference	80
5.4.1	Detailed Description	81
5.4.2	Macro Definition Documentation	81
5.4.2.1	LSLOGGING_FILE_NAME	81
5.4.2.2	LSLOGGING_MSG_LENGTH	81
5.4.2.3	LSLOGGING_QUEUE_LENGTH	81
5.4.3	Typedef Documentation	82
5.4.3.1	lslogging_queue_t	82
5.4.4	Function Documentation	82
5.4.4.1	lslogging_init	82
5.4.4.2	lslogging_log_message	82
5.4.4.3	lslogging_run	82
5.4.4.4	lslogging_worker	83
5.4.5	Variable Documentation	83
5.4.5.1	lslogging_cond	83
5.4.5.2	lslogging_file	83
5.4.5.3	lslogging_mutex	83
5.4.5.4	lslogging_off	84
5.4.5.5	lslogging_on	84
5.4.5.6	lslogging_queue	84
5.4.5.7	lslogging_thread	84
5.5	lspg.c File Reference	84
5.5.1	Detailed Description	88
5.5.2	Macro Definition Documentation	89
5.5.2.1	LS_PG_QUERY_QUEUE_LENGTH	89
5.5.2.2	LS_PG_STATE_IDLE	89
5.5.2.3	LS_PG_STATE_INIT	89
5.5.2.4	LS_PG_STATE_INIT_POLL	89
5.5.2.5	LS_PG_STATE_RECV	89
5.5.2.6	LS_PG_STATE_RESET	89
5.5.2.7	LS_PG_STATE_RESET_POLL	89
5.5.2.8	LS_PG_STATE_SEND	89
5.5.2.9	LS_PG_STATE_SEND_FLUSH	89
5.5.3	Typedef Documentation	89
5.5.3.1	lspg_lock_detector_t	89
5.5.3.2	lspg_lock_diffractionmeter_t	89
5.5.3.3	lspg_query_queue_t	90
5.5.3.4	lspg_seq_run_prep_t	90

5.5.3.5	lspg_wait_for_detector_t	90
5.5.4	Function Documentation	90
5.5.4.1	lspg_array2ptrs	90
5.5.4.2	lspg_blight_lut_cb	91
5.5.4.3	lspg_cmd_cb	92
5.5.4.4	lspg_flight_lut_cb	92
5.5.4.5	lspg_flush	93
5.5.4.6	lspg_getcenter_all	93
5.5.4.7	lspg_getcenter_call	93
5.5.4.8	lspg_getcenter_cb	94
5.5.4.9	lspg_getcenter_done	95
5.5.4.10	lspg_getcenter_init	95
5.5.4.11	lspg_getcenter_wait	95
5.5.4.12	lspg_init	95
5.5.4.13	lspg_init_motors_cb	95
5.5.4.14	lspg_kvs_cb	96
5.5.4.15	lspg_lock_detector_all	97
5.5.4.16	lspg_lock_detector_call	97
5.5.4.17	lspg_lock_detector_cb	97
5.5.4.18	lspg_lock_detector_done	98
5.5.4.19	lspg_lock_detector_init	98
5.5.4.20	lspg_lock_detector_wait	98
5.5.4.21	lspg_lock_diffractionmeter_all	98
5.5.4.22	lspg_lock_diffractionmeter_call	98
5.5.4.23	lspg_lock_diffractionmeter_cb	99
5.5.4.24	lspg_lock_diffractionmeter_done	99
5.5.4.25	lspg_lock_diffractionmeter_init	99
5.5.4.26	lspg_lock_diffractionmeter_wait	99
5.5.4.27	lspg_next_state	99
5.5.4.28	lspg_nextaction_cb	100
5.5.4.29	lspg_nextshot_call	101
5.5.4.30	lspg_nextshot_cb	101
5.5.4.31	lspg_nextshot_done	105
5.5.4.32	lspg_nextshot_init	105
5.5.4.33	lspg_nextshot_wait	105
5.5.4.34	lspg_notice_processor	105
5.5.4.35	lspg_pg_connect	105
5.5.4.36	lspg_pg_service	107
5.5.4.37	lspg_query_next	108
5.5.4.38	lspg_query_push	109

5.5.4.39	<code>lspg_query_reply_next</code>	109
5.5.4.40	<code>lspg_query_reply_peek</code>	109
5.5.4.41	<code>lspg_receive</code>	110
5.5.4.42	<code>lspg_run</code>	111
5.5.4.43	<code>lspg_scint_lut_cb</code>	111
5.5.4.44	<code>lspg_send_next_query</code>	111
5.5.4.45	<code>lspg_seq_run_prep_all</code>	112
5.5.4.46	<code>lspg_seq_run_prep_call</code>	112
5.5.4.47	<code>lspg_seq_run_prep_cb</code>	113
5.5.4.48	<code>lspg_seq_run_prep_done</code>	113
5.5.4.49	<code>lspg_seq_run_prep_init</code>	113
5.5.4.50	<code>lspg_seq_run_prep_wait</code>	113
5.5.4.51	<code>lspg_sig_service</code>	114
5.5.4.52	<code>lspg_wait_for_detector_all</code>	114
5.5.4.53	<code>lspg_wait_for_detector_call</code>	114
5.5.4.54	<code>lspg_wait_for_detector_cb</code>	115
5.5.4.55	<code>lspg_wait_for_detector_done</code>	115
5.5.4.56	<code>lspg_wait_for_detector_init</code>	115
5.5.4.57	<code>lspg_wait_for_detector_wait</code>	115
5.5.4.58	<code>lspg_worker</code>	115
5.5.4.59	<code>lspg_zoom_lut_cb</code>	117
5.5.5	Variable Documentation	117
5.5.5.1	<code>ls_pg_state</code>	117
5.5.5.2	<code>lspg_connectPoll_response</code>	117
5.5.5.3	<code>lspg_getcenter</code>	117
5.5.5.4	<code>lspg_lock_detector</code>	117
5.5.5.5	<code>lspg_lock_diffractionmeter</code>	117
5.5.5.6	<code>lspg_nextshot</code>	118
5.5.5.7	<code>lspg_query_queue</code>	118
5.5.5.8	<code>lspg_query_queue_off</code>	118
5.5.5.9	<code>lspg_query_queue_on</code>	118
5.5.5.10	<code>lspg_query_queue_reply</code>	118
5.5.5.11	<code>lspg_queue_cond</code>	118
5.5.5.12	<code>lspg_queue_mutex</code>	118
5.5.5.13	<code>lspg_resetPoll_response</code>	118
5.5.5.14	<code>lspg_seq_run_prep</code>	118
5.5.5.15	<code>lspg_thread</code>	119
5.5.5.16	<code>lspg_wait_for_detector</code>	119
5.5.5.17	<code>lspgfd</code>	119
5.5.5.18	<code>now</code>	119

5.5.5.19	q	119
5.6	lspmac.c File Reference	119
5.6.1	Detailed Description	126
5.6.2	Macro Definition Documentation	126
5.6.2.1	LS_PMAC_STATE_CR	126
5.6.2.2	LS_PMAC_STATE_DETACHED	126
5.6.2.3	LS_PMAC_STATE_GB	126
5.6.2.4	LS_PMAC_STATE_GMR	127
5.6.2.5	LS_PMAC_STATE_IDLE	127
5.6.2.6	LS_PMAC_STATE_RESET	127
5.6.2.7	LS_PMAC_STATE_RR	127
5.6.2.8	LS_PMAC_STATE_SC	127
5.6.2.9	LS_PMAC_STATE_WACK	127
5.6.2.10	LS_PMAC_STATE_WACK_CC	127
5.6.2.11	LS_PMAC_STATE_WACK_NFR	127
5.6.2.12	LS_PMAC_STATE_WACK_RR	127
5.6.2.13	LS_PMAC_STATE_WCR	127
5.6.2.14	LS_PMAC_STATE_WGB	127
5.6.2.15	LSPMAC_PRESET_REGEX	127
5.6.2.16	PMAC_CMD_QUEUE_LENGTH	128
5.6.2.17	pmac_cmd_size	128
5.6.2.18	PMAC_MIN_CMD_TIME	128
5.6.2.19	PMACPORT	128
5.6.2.20	VR_CTRL_RESPONSE	128
5.6.2.21	VR_DOWNLOAD	128
5.6.2.22	VR_FWDOWNLOAD	128
5.6.2.23	VR_IPADDRESS	128
5.6.2.24	VR_PMAC_FLUSH	128
5.6.2.25	VR_PMAC_GETBUFFER	128
5.6.2.26	VR_PMAC_GETLINE	128
5.6.2.27	VR_PMAC_GETMEM	129
5.6.2.28	VR_PMAC_GETRESPONSE	129
5.6.2.29	VR_PMAC_PORT	129
5.6.2.30	VR_PMAC_READREADY	129
5.6.2.31	VR_PMAC_SENDCTRLCHAR	129
5.6.2.32	VR_PMAC_SENDLINE	129
5.6.2.33	VR_PMAC_SETBIT	129
5.6.2.34	VR_PMAC_SETBITS	129
5.6.2.35	VR_PMAC_SETMEM	129
5.6.2.36	VR_PMAC_WRITEBUFFER	129

5.6.2.37	VR_PMAC_WRITEERROR	129
5.6.2.38	VR_UPLOAD	129
5.6.3	Typedef Documentation	130
5.6.3.1	md2_status_t	130
5.6.4	Function Documentation	130
5.6.4.1	cleanstr	130
5.6.4.2	hex_dump	130
5.6.4.3	IsConnect	131
5.6.4.4	lspmac_backLight_down_cb	131
5.6.4.5	lspmac_backLight_up_cb	132
5.6.4.6	lspmac_bi_init	132
5.6.4.7	lspmac_bo_init	132
5.6.4.8	lspmac_bo_read	133
5.6.4.9	lspmac_cryoSwitchChanged_cb	133
5.6.4.10	lspmac_dac_init	133
5.6.4.11	lspmac_dac_read	134
5.6.4.12	lspmac_Error	135
5.6.4.13	lspmac_fshut_init	135
5.6.4.14	lspmac_get_status	136
5.6.4.15	lspmac_get_status_cb	136
5.6.4.16	lspmac_GetAllIVars	138
5.6.4.17	lspmac_GetAllIVarsCB	138
5.6.4.18	lspmac_GetAllMVars	139
5.6.4.19	lspmac_GetAllMVarsCB	139
5.6.4.20	lspmac_Getmem	139
5.6.4.21	lspmac_GetmemReplyCB	139
5.6.4.22	lspmac_getPosition	140
5.6.4.23	lspmac_GetShortReplyCB	140
5.6.4.24	lspmac_home1_queue	141
5.6.4.25	lspmac_home2_queue	142
5.6.4.26	lspmac_init	142
5.6.4.27	lspmac_jogabs_queue	144
5.6.4.28	lspmac_light_zoom_cb	145
5.6.4.29	lspmac_lut	145
5.6.4.30	lspmac_motor_init	146
5.6.4.31	lspmac_move_or_jog_abs_queue	147
5.6.4.32	lspmac_move_or_jog_preset_queue	149
5.6.4.33	lspmac_move_preset_queue	149
5.6.4.34	lspmac_moveabs_blight_factor_queue	150
5.6.4.35	lspmac_moveabs_bo_queue	150

5.6.4.36	lspmac_moveabs_flight_factor_queue	151
5.6.4.37	lspmac_moveabs_frontlight_oo_queue	151
5.6.4.38	lspmac_moveabs_fshut_queue	151
5.6.4.39	lspmac_moveabs_queue	152
5.6.4.40	lspmac_moveabs_timed_queue	152
5.6.4.41	lspmac_moveabs_wait	153
5.6.4.42	lspmac_movedac_queue	154
5.6.4.43	lspmac_movezoom_queue	155
5.6.4.44	lspmac_newKV_cb	155
5.6.4.45	lspmac_next_state	156
5.6.4.46	lspmac_pmacmotor_read	157
5.6.4.47	lspmac_pop_queue	160
5.6.4.48	lspmac_pop_reply	160
5.6.4.49	lspmac_push_queue	161
5.6.4.50	lspmac_Reset	161
5.6.4.51	lspmac_rlut	161
5.6.4.52	lspmac_run	162
5.6.4.53	lspmac_scint_dried_cb	162
5.6.4.54	lspmac_scint_inPosition_cb	163
5.6.4.55	lspmac_send_command	163
5.6.4.56	lspmac_sendcmd	164
5.6.4.57	lspmac_sendcmd_nocb	164
5.6.4.58	lspmac_SendControlReplyPrintCB	165
5.6.4.59	lspmac_Service	165
5.6.4.60	lspmac_shutter_read	167
5.6.4.61	lspmac_SockFlush	168
5.6.4.62	lspmac_SockGetmem	168
5.6.4.63	lspmac_SockSendControlCharPrint	169
5.6.4.64	lspmac_SockSendline	169
5.6.4.65	lspmac_SockSendline_nr	169
5.6.4.66	lspmac_soft_motor_init	170
5.6.4.67	lspmac_soft_motor_read	170
5.6.4.68	lspmac_video_rotate	170
5.6.4.69	lspmac_worker	171
5.6.5	Variable Documentation	171
5.6.5.1	alignx	171
5.6.5.2	aligny	171
5.6.5.3	alignz	171
5.6.5.4	anal	172
5.6.5.5	apery	172

5.6.5.6	aperz	172
5.6.5.7	blight	172
5.6.5.8	blight_f	172
5.6.5.9	blight_ud	172
5.6.5.10	capy	172
5.6.5.11	capz	172
5.6.5.12	cenx	172
5.6.5.13	ceny	173
5.6.5.14	cr_cmd	173
5.6.5.15	cryo	173
5.6.5.16	cryo_switch	173
5.6.5.17	dbmem	173
5.6.5.18	dbmemIn	173
5.6.5.19	dryer	173
5.6.5.20	ethCmdOff	173
5.6.5.21	ethCmdOn	173
5.6.5.22	ethCmdQueue	174
5.6.5.23	ethCmdReply	174
5.6.5.24	flight	174
5.6.5.25	flight_f	174
5.6.5.26	flight_oo	174
5.6.5.27	fluo	174
5.6.5.28	fscint	174
5.6.5.29	fshut	174
5.6.5.30	gb_cmd	174
5.6.5.31	getivars	175
5.6.5.32	getmvars	175
5.6.5.33	kappa	175
5.6.5.34	linesReceived	175
5.6.5.35	ls_pmac_state	175
5.6.5.36	lspmac_bis	175
5.6.5.37	lspmac_motors	175
5.6.5.38	lspmac_moving_cond	175
5.6.5.39	lspmac_moving_flags	175
5.6.5.40	lspmac_moving_mutex	176
5.6.5.41	lspmac_nbis	176
5.6.5.42	lspmac_nmotors	176
5.6.5.43	lspmac_shutter_cond	176
5.6.5.44	lspmac_shutter_has_opened	176
5.6.5.45	lspmac_shutter_mutex	176

5.6.5.46	lspmac_shutter_state	176
5.6.5.47	lspmac_status_last_time	176
5.6.5.48	lspmac_status_time	176
5.6.5.49	md2_status	177
5.6.5.50	md2_status_mutex	177
5.6.5.51	now	177
5.6.5.52	omega	177
5.6.5.53	omega_zero_search	177
5.6.5.54	omega_zero_time	177
5.6.5.55	omega_zero_velocity	177
5.6.5.56	phi	177
5.6.5.57	pmac_error_strs	177
5.6.5.58	pmac_queue_cond	178
5.6.5.59	pmac_queue_mutex	178
5.6.5.60	pmac_thread	178
5.6.5.61	pmacfd	178
5.6.5.62	rr_cmd	178
5.6.5.63	scint	178
5.6.5.64	zoom	178
5.7	Isredis.c File Reference	179
5.7.1	Detailed Description	180
5.7.2	Function Documentation	180
5.7.2.1	_Isredis_get_obj	180
5.7.2.2	_Isredis_set_value	181
5.7.2.3	Isredis_addRead	182
5.7.2.4	Isredis_addWrite	182
5.7.2.5	Isredis_cleanup	182
5.7.2.6	Isredis_debugCB	182
5.7.2.7	Isredis_delRead	183
5.7.2.8	Isredis_delWrite	183
5.7.2.9	Isredis_fd_service	184
5.7.2.10	Isredis_get_obj	184
5.7.2.11	Isredis_getd	184
5.7.2.12	Isredis_getl	185
5.7.2.13	Isredis_getstr	185
5.7.2.14	Isredis_hgetCB	185
5.7.2.15	Isredis_init	186
5.7.2.16	Isredis_isvalid	186
5.7.2.17	Isredis_keysCB	187
5.7.2.18	Isredis_maybe_add_key	187

5.7.2.19	lsredis_run	187
5.7.2.20	lsredis_select	187
5.7.2.21	lsredis_set_invalid	188
5.7.2.22	lsredis_set_value	188
5.7.2.23	lsredis_setstr	188
5.7.2.24	lsredis_subCB	189
5.7.2.25	lsredis_worker	191
5.7.2.26	redisDisconnectCB	192
5.7.3	Variable Documentation	192
5.7.3.1	lsredis_head	192
5.7.3.2	lsredis_key_select_regex	192
5.7.3.3	lsredis_objs	192
5.7.3.4	lsredis_objs_mutex	192
5.7.3.5	lsredis_publisher	192
5.7.3.6	lsredis_ro_mutex	192
5.7.3.7	lsredis_thread	192
5.7.3.8	lsredis_wr_mutex	192
5.7.3.9	roac	192
5.7.3.10	rofd	193
5.7.3.11	subac	193
5.7.3.12	subfd	193
5.7.3.13	wrac	193
5.7.3.14	wrfd	193
5.8	lstimer.c File Reference	193
5.8.1	Detailed Description	194
5.8.2	Macro Definition Documentation	194
5.8.2.1	LSTIMER_LIST_LENGTH	194
5.8.2.2	LSTIMER_RESOLUTION_NSECS	195
5.8.3	Typedef Documentation	195
5.8.3.1	lstimer_list_t	195
5.8.4	Function Documentation	195
5.8.4.1	handler	195
5.8.4.2	lstimer_add_timer	195
5.8.4.3	lstimer_init	196
5.8.4.4	lstimer_run	196
5.8.4.5	lstimer_worker	196
5.8.4.6	service_timers	197
5.8.5	Variable Documentation	198
5.8.5.1	lstimer_active_timers	198
5.8.5.2	lstimer_cond	199

5.8.5.3	lstimer_list	199
5.8.5.4	lstimer_mutex	199
5.8.5.5	lstimer_thread	199
5.8.5.6	lstimer_timerid	199
5.8.5.7	new_timer	199
5.9	lsupdate.c File Reference	199
5.9.1	Detailed Description	200
5.9.2	Function Documentation	200
5.9.2.1	lsupdate_init	200
5.9.2.2	lsupdate_run	200
5.9.2.3	lsupdate_updateit	200
5.9.2.4	lsupdate_worker	202
5.9.3	Variable Documentation	202
5.9.3.1	lsupdate_thread	202
5.10	md2cmds.c File Reference	202
5.10.1	Detailed Description	203
5.10.2	Function Documentation	204
5.10.2.1	md2cmds_center	204
5.10.2.2	md2cmds_collect	204
5.10.2.3	md2cmds_init	206
5.10.2.4	md2cmds_maybe_done_moving_cb	206
5.10.2.5	md2cmds_maybe_rotate_done_cb	207
5.10.2.6	md2cmds_moveAbs	207
5.10.2.7	md2cmds_mvcenter_move	208
5.10.2.8	md2cmds_mvcenter_prep	209
5.10.2.9	md2cmds_mvcenter_wait	210
5.10.2.10	md2cmds_phase_change	210
5.10.2.11	md2cmds_prep_motion	212
5.10.2.12	md2cmds_rotate	213
5.10.2.13	md2cmds_rotate_cb	214
5.10.2.14	md2cmds_run	214
5.10.2.15	md2cmds_set_scale_cb	215
5.10.2.16	md2cmds_transfer	215
5.10.2.17	md2cmds_worker	215
5.10.3	Variable Documentation	216
5.10.3.1	md2cmds_cmd	216
5.10.3.2	md2cmds_cond	216
5.10.3.3	md2cmds_moving_cond	216
5.10.3.4	md2cmds_moving_count	216
5.10.3.5	md2cmds_moving_mutex	216

5.10.3.6	md2cmds_moving_pq	216
5.10.3.7	md2cmds_mutex	216
5.10.3.8	md2cmds_thread	217
5.10.3.9	rotating	217
5.11	pgpmac.c File Reference	217
5.11.1	Detailed Description	217
5.11.2	Function Documentation	218
5.11.2.1	main	218
5.11.2.2	pgpmac_printf	219
5.11.2.3	stdinService	220
5.11.3	Variable Documentation	221
5.11.3.1	ncurses_mutex	221
5.11.3.2	stdinfda	221
5.11.3.3	term_input	221
5.11.3.4	term_output	221
5.11.3.5	term_status	221
5.11.3.6	term_status2	221
5.12	pgpmac.h File Reference	222
5.12.1	Detailed Description	227
5.12.2	Macro Definition Documentation	227
5.12.2.1	LS_DISPLAY_WINDOW_HEIGHT	227
5.12.2.2	LS_DISPLAY_WINDOW_WIDTH	227
5.12.2.3	LS_PG_QUERY_STRING_LENGTH	227
5.12.2.4	LSEVENTS_EVENT_LENGTH	227
5.12.2.5	MD2CMD5_CMD_LENGTH	227
5.12.3	Typedef Documentation	227
5.12.3.1	lskvs_kvs_list_t	227
5.12.3.2	lskvs_kvs_t	228
5.12.3.3	lspg_getcenter_t	228
5.12.3.4	lspg_nextshot_t	228
5.12.3.5	lspmac_bi_t	228
5.12.3.6	lspmac_motor_t	228
5.12.3.7	lsredis_obj_t	228
5.12.3.8	pmac_cmd_queue_t	228
5.12.3.9	pmac_cmd_t	228
5.12.4	Function Documentation	228
5.12.4.1	lsevents_add_listener	228
5.12.4.2	lsevents_init	229
5.12.4.3	lsevents_remove_listener	229
5.12.4.4	lsevents_run	230

5.12.4.5	<code>lsevents_send_event</code>	230
5.12.4.6	<code>lskvs_find_preset_position</code>	231
5.12.4.7	<code>lskvs_regcomp</code>	232
5.12.4.8	<code>lspg_init</code>	233
5.12.4.9	<code>lspg_run</code>	233
5.12.4.10	<code>lspg_seq_run_prep_all</code>	233
5.12.4.11	<code>lspg_zoom_lut_call</code>	234
5.12.4.12	<code>lspmac_getPosition</code>	234
5.12.4.13	<code>lspmac_init</code>	234
5.12.4.14	<code>lspmac_jogabs_queue</code>	236
5.12.4.15	<code>lspmac_move_or_jog_preset_queue</code>	236
5.12.4.16	<code>lspmac_move_or_jog_queue</code>	236
5.12.4.17	<code>lspmac_moveabs_queue</code>	236
5.12.4.18	<code>lspmac_run</code>	237
5.12.4.19	<code>lspmac_SockSendline</code>	237
5.12.4.20	<code>lsredis_get_obj</code>	237
5.12.4.21	<code>lsredis_init</code>	238
5.12.4.22	<code>lsredis_run</code>	239
5.12.4.23	<code>lstimer_add_timer</code>	239
5.12.4.24	<code>lstimer_init</code>	240
5.12.4.25	<code>lstimer_run</code>	240
5.12.4.26	<code>lsupdate_init</code>	240
5.12.4.27	<code>lsupdate_run</code>	240
5.12.4.28	<code>md2cmds_init</code>	240
5.12.4.29	<code>md2cmds_run</code>	241
5.12.4.30	<code>pgpmac_printf</code>	241
5.12.4.31	<code>PmacSockSendline</code>	241
5.12.5	Variable Documentation	241
5.12.5.1	<code>alignx</code>	241
5.12.5.2	<code>aligny</code>	242
5.12.5.3	<code>alignz</code>	242
5.12.5.4	<code>anal</code>	242
5.12.5.5	<code>apery</code>	242
5.12.5.6	<code>aperz</code>	242
5.12.5.7	<code>blight</code>	242
5.12.5.8	<code>blight_f</code>	242
5.12.5.9	<code>blight_ud</code>	242
5.12.5.10	<code>capy</code>	242
5.12.5.11	<code>capz</code>	243
5.12.5.12	<code>cenx</code>	243

5.12.5.13 ceny	243
5.12.5.14 cryo	243
5.12.5.15 dryer	243
5.12.5.16 flight	243
5.12.5.17 flight_f	243
5.12.5.18 flight_oo	243
5.12.5.19 fluo	243
5.12.5.20 fscint	244
5.12.5.21 fshut	244
5.12.5.22 kappa	244
5.12.5.23 lskvs_kvs	244
5.12.5.24 lskvs_rwlock	244
5.12.5.25 lspg_getcenter	244
5.12.5.26 lspg_nextshot	244
5.12.5.27 lspmac_motors	244
5.12.5.28 lspmac_moving_cond	244
5.12.5.29 lspmac_moving_flags	245
5.12.5.30 lspmac_moving_mutex	245
5.12.5.31 lspmac_nmotors	245
5.12.5.32 lspmac_shutter_cond	245
5.12.5.33 lspmac_shutter_has_opened	245
5.12.5.34 lspmac_shutter_mutex	245
5.12.5.35 lspmac_shutter_state	245
5.12.5.36 md2_status_mutex	245
5.12.5.37 md2cmds_cmd	245
5.12.5.38 md2cmds_cond	246
5.12.5.39 md2cmds_mutex	246
5.12.5.40 md2cmds_pg_cond	246
5.12.5.41 md2cmds_pg_mutex	246
5.12.5.42 ncurses_mutex	246
5.12.5.43 omega	246
5.12.5.44 omega_zero_time	246
5.12.5.45 phi	246
5.12.5.46 pmac_queue_cond	246
5.12.5.47 pmac_queue_mutex	246
5.12.5.48 scint	246
5.12.5.49 term_input	247
5.12.5.50 term_output	247
5.12.5.51 term_status	247
5.12.5.52 term_status2	247

5.12.5.53 zoom	247
--------------------------	-----

Index	247
--------------	------------

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:

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

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

lsevents_listener_struct	Linked list of event listeners	9
lsevents_queue_struct	Storage definition for the events	10
lskvs_kvs_list_struct	A second linked list type to handle private lists of KVs	10
lskvs_kvs_struct	Storage for the key value pairs	11
lslogging_queue_struct	Our log object: time and message	12
lspg_getcenter_struct	Storage for getcenter query Used for the md2 ROTATE command that generates the centering movies	13
lspg_lock_detector_struct	Lock detector object Implements detector lock for exposure control	15
lspg_lock_diffractionmeter_struct	Object used to impliment locking the diffractometer Critical to exposure timing	16
lspg_nextshot_struct	Storage definition for nextshot query	17
lspg_seq_run_prep_struct	Data collection running object	28
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	28
lspgQueryQueueStruct	Store each query along with it's callback function	29
lspmac_bi_struct	Storage for binary inputs	30
lspmac_cmd_queue_struct	PMAC command queue item	31
lspmac_motor_struct	Motor information	32
lsredis_obj_struct	Redis Object Basic object whose value is synchronized with our redis db	39
lstimer_list_struct	Everything we need to know about a timer	41
md2StatusStruct	The block of memory retrieved in a status request	43

tagEthernetCmd	
PMAC ethernet packet definition	50

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

kvredis.c	53
lsevents.c	
Event subsystem for inter-pgpmac communication	69
lskvs.c	
Support for the remote access client key value pairs	74
lslogging.c	
Logs messages to a file	80
lspg.c	
Postgresql support for the LS-CAT pgpmac project	84
lspmac.c	
Routines concerned with communication with PMAC	119
lsredis.c	
Support redis hash synchronization	179
lstimer.c	
Support for delayed and periodic events	193
lsupdate.c	
Brings this MD2 code and the database kvs table into agreement	199
md2cmds.c	
Implements commands to run the md2 diffractometer attached to a PMAC controled by post-gresql	202
pgpmac.c	
Main for the pgpmac project	217
pgpmac.h	
Headers for the entire pgpmac project	222

Chapter 4

Data Structure Documentation

4.1 lsevents_listener_struct Struct Reference

Linked list of event listeners.

Data Fields

- struct lsevents_listener_struct * next
Next listener.
- char * raw_regexp
the original string sent to us
- regex_t re
regular expression representing listened for events
- void(* cb)(char *)
call back function

4.1.1 Detailed Description

Linked list of event listeners.

Definition at line 27 of file lsevents.c.

4.1.2 Field Documentation

4.1.2.1 void(* lsevents_listener_struct::cb)(char *)

call back function

Definition at line 31 of file lsevents.c.

4.1.2.2 struct lsevents_listener_struct* lsevents_listener_struct::next

Next listener.

Definition at line 28 of file lsevents.c.

4.1.2.3 `char* lsevents_listener_struct::raw_regexp`

the original string sent to us

Definition at line 29 of file lsevents.c.

4.1.2.4 `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](#)

4.2 lsevents_queue_struct Struct Reference

Storage definition for the events.

Data Fields

- `char event [LSEVENTS_EVENT_LENGTH]`
name of the event

4.2.1 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.

4.2.2 Field Documentation

4.2.2.1 `char lsevents_queue_struct::event[LSEVENTS_EVENT_LENGTH]`

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](#)

4.3 lskvs_kvs_list_struct Struct Reference

A second linked list type to handle private lists of KVs.

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

Data Fields

- `struct lskvs_kvs_list_struct * next`

next item

- [lskvs_kvs_t](#) * *kvs*

the KV

4.3.1 Detailed Description

A second linked list type to handle private lists of KVs.

Developed to support lists of preset motor positions.

Definition at line 106 of file pgpmac.h.

4.3.2 Field Documentation

4.3.2.1 [lskvs_kvs_t](#)* [lskvs_kvs_list_struct::kvs](#)

the KV

Definition at line 108 of file pgpmac.h.

4.3.2.2 [struct lskvs_kvs_list_struct](#)* [lskvs_kvs_list_struct::next](#)

next item

Definition at line 107 of file pgpmac.h.

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

- [pgpmac.h](#)

4.4 lskvs_kvs_struct Struct Reference

Storage for the key value pairs.

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

Data Fields

- [struct lskvs_kvs_struct](#) * *next*

the next kvpair

- [pthread_rwlock_t](#) *l*

our lock

- [char](#) * *k*

the key

- [char](#) * *v*

the value

- [int](#) *vl*

the length of the allocated v

4.4.1 Detailed Description

Storage for the key value pairs.

the k's and v's are strings and to keep the memory management less crazy we'll calloc some space for these strings and only free and re-calloc if we need more space later. Only the values are ever going to be resized.

Definition at line 95 of file pgpmac.h.

4.4.2 Field Documentation

4.4.2.1 `char* lskvs_kvs_struct::k`

the key

Definition at line 98 of file pgpmac.h.

4.4.2.2 `pthread_rwlock_t lskvs_kvs_struct::l`

our lock

Definition at line 97 of file pgpmac.h.

4.4.2.3 `struct lskvs_kvs_struct* lskvs_kvs_struct::next`

the next kvpair

Definition at line 96 of file pgpmac.h.

4.4.2.4 `char* lskvs_kvs_struct::v`

the value

Definition at line 99 of file pgpmac.h.

4.4.2.5 `int lskvs_kvs_struct::vl`

the length of the calloced v

Definition at line 100 of file pgpmac.h.

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

- [pgpmac.h](#)

4.5 Islogging_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

4.5.1 Detailed Description

Our log object: time and message.

Definition at line 24 of file lslogging.c.

4.5.2 Field Documentation

4.5.2.1 char lslogging_queue_struct::lmsg[LSLOGGING_MSG_LENGTH]

our message, truncated if too long

Definition at line 26 of file lslogging.c.

4.5.2.2 struct timespec lslogging_queue_struct::ltime

time stamp: set when queued

Definition at line 25 of file lslogging.c.

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

- [lslogging.c](#)

4.6 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](#)

4.6.1 Detailed Description

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

Definition at line 181 of file pgpmac.h.

4.6.2 Field Documentation

4.6.2.1 pthread_cond_t lspg_getcenter_struct::cond

provides signaling for when the query is done

Definition at line 183 of file pgpmac.h.

4.6.2.2 double lspg_getcenter_struct::dax

alignment x change

Definition at line 196 of file pgpmac.h.

4.6.2.3 int lspg_getcenter_struct::dax_isnull

Definition at line 197 of file pgpmac.h.

4.6.2.4 double lspg_getcenter_struct::day

alignment y change

Definition at line 199 of file pgpmac.h.

4.6.2.5 int lspg_getcenter_struct::day_isnull

Definition at line 200 of file pgpmac.h.

4.6.2.6 double lspg_getcenter_struct::daz

alignment z change

Definition at line 202 of file pgpmac.h.

4.6.2.7 int lspg_getcenter_struct::daz_isnull

Definition at line 203 of file pgpmac.h.

4.6.2.8 double lspg_getcenter_struct::dcx

center x change

Definition at line 190 of file pgpmac.h.

4.6.2.9 int lspg_getcenter_struct::dcx_isnull

Definition at line 191 of file pgpmac.h.

4.6.2.10 double lspg_getcenter_struct::dcy

center y change

Definition at line 193 of file pgpmac.h.

4.6.2.11 int lspg_getcenter_struct::dcy_isnull

Definition at line 194 of file pgpmac.h.

4.6.2.12 pthread_mutex_t lspg_getcenter_struct::mutex

don't let the threads collide!

Definition at line 182 of file pgpmac.h.

4.6.2.13 int lspg_getcenter_struct::new_value_ready

used with condition

Definition at line 184 of file pgpmac.h.

4.6.2.14 int lspg_getcenter_struct::no_rows_returned

flag in case no centering information was forthcoming

Definition at line 185 of file pgpmac.h.

4.6.2.15 int lspg_getcenter_struct::zoom

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

Definition at line 187 of file pgpmac.h.

4.6.2.16 int lspg_getcenter_struct::zoom_isnull

Definition at line 188 of file pgpmac.h.

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

- [pgpmac.h](#)

4.7 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](#)

4.7.1 Detailed Description

lock detector object Implements detector lock for exposure control

Definition at line 865 of file lspg.c.

4.7.2 Field Documentation

4.7.2.1 pthread_cond_t lspg_lock_detector_struct::cond

Definition at line 867 of file lspg.c.

4.7.2.2 pthread_mutex_t lspg_lock_detector_struct::mutex

Definition at line 866 of file lspg.c.

4.7.2.3 int lspg_lock_detector_struct::new_value_ready

Definition at line 868 of file lspg.c.

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

- [lspg.c](#)

4.8 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](#)

4.8.1 Detailed Description

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

Definition at line 806 of file lspg.c.

4.8.2 Field Documentation

4.8.2.1 pthread_cond_t lspg_lock_diffractionmeter_struct::cond

Definition at line 808 of file lspg.c.

4.8.2.2 pthread_mutex_t lspg_lock_diffractionmeter_struct::mutex

Definition at line 807 of file lspg.c.

4.8.2.3 int lspg_lock_diffractionmeter_struct::new_value_ready

Definition at line 809 of file lspg.c.

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

- [lspg.c](#)

4.9 lspg_nextshot_struct Struct Reference

Storage definition for nextshot query.

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

Data Fields

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

- double `dsphi`
dataset defined starting phi angle
- int `dsphi_isnull`
- double `dsomega`
dataset defined starting omega angle
- int `dsomega_isnull`
- double `dskappa`
dataset defined starting kappa angle
- int `dskappa_isnull`
- double `dsdist`
dataset defined detector distance
- int `dsdist_isnull`
- double `dsnrg`
dataset defined energy
- int `dsnrg_isnull`
- unsigned int `dshpid`
sample holder ID
- int `dshpid_isnull`
- double `cx`
centering table x position
- int `cx_isnull`
- double `cy`
centering table y position
- int `cy_isnull`
- double `ax`
alignment table x position
- int `ax_isnull`
- double `ay`
alignment table y position
- int `ay_isnull`
- double `az`
alignment table z position
- int `az_isnull`
- int `active`
flag: 1=move to indicated center position, 0=don't move center or alignment tables
- int `active_isnull`
- int `sindex`
index of frame (used to generate the file extension)
- int `sindex_isnull`
- char * `stype`
"Normal" or "Gridsearch"
- int `stype_isnull`
- double `dsowidth2`
next image oscillation width
- int `dsowidth2_isnull`
- char * `dsoscaxis2`
next image oscillation axis (always "omega")
- int `dsoscaxis2_isnull`
- double `dsexp2`
next image exposure time
- int `dsexp2_isnull`
- double `sstart2`

- next image start angle*
 - int [sstart2_isnull](#)
 - double [dspphi2](#)
- next image phi position*
 - int [dspphi2_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](#)

4.9.1 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 217 of file pgpmac.h.

4.9.2 Field Documentation

4.9.2.1 `int lspg_nextshot_struct::active`

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

Definition at line 280 of file pgpmac.h.

4.9.2.2 `int lspg_nextshot_struct::active2`

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

Definition at line 331 of file pgpmac.h.

4.9.2.3 `int lspg_nextshot_struct::active2_isnull`

Definition at line 332 of file pgpmac.h.

4.9.2.4 `int lspg_nextshot_struct::active_isnull`

Definition at line 281 of file pgpmac.h.

4.9.2.5 `double lspg_nextshot_struct::ax`

alignment table x position

Definition at line 271 of file pgpmac.h.

4.9.2.6 `double lspg_nextshot_struct::ax2`

next image alignment x position

Definition at line 322 of file pgpmac.h.

4.9.2.7 `int lspg_nextshot_struct::ax2_isnull`

Definition at line 323 of file pgpmac.h.

4.9.2.8 `int lspg_nextshot_struct::ax_isnull`

Definition at line 272 of file pgpmac.h.

4.9.2.9 `double lspg_nextshot_struct::ay`

alignment table y position

Definition at line 274 of file pgpmac.h.

4.9.2.10 `double lspg_nextshot_struct::ay2`

next image alignment y position

Definition at line 325 of file pgpmac.h.

4.9.2.11 int lspg_nextshot_struct::ay2_isnull

Definition at line 326 of file pgpmac.h.

4.9.2.12 int lspg_nextshot_struct::ay_isnull

Definition at line 275 of file pgpmac.h.

4.9.2.13 double lspg_nextshot_struct::az

alignment table z position

Definition at line 277 of file pgpmac.h.

4.9.2.14 double lspg_nextshot_struct::az2

next image alignment z position

Definition at line 328 of file pgpmac.h.

4.9.2.15 int lspg_nextshot_struct::az2_isnull

Definition at line 329 of file pgpmac.h.

4.9.2.16 int lspg_nextshot_struct::az_isnull

Definition at line 278 of file pgpmac.h.

4.9.2.17 pthread_cond_t lspg_nextshot_struct::cond

Condition to wait for a response from our postgresql server.

Definition at line 219 of file pgpmac.h.

4.9.2.18 double lspg_nextshot_struct::cx

centering table x position

Definition at line 265 of file pgpmac.h.

4.9.2.19 double lspg_nextshot_struct::cx2

next image centering table x position

Definition at line 316 of file pgpmac.h.

4.9.2.20 int lspg_nextshot_struct::cx2_isnull

Definition at line 317 of file pgpmac.h.

4.9.2.21 int lspg_nextshot_struct::cx_isnull

Definition at line 266 of file pgpmac.h.

4.9.2.22 `double lspg_nextshot_struct::cy`

centering table y position

Definition at line 268 of file pgpmac.h.

4.9.2.23 `double lspg_nextshot_struct::cy2`

next image centering table y position

Definition at line 319 of file pgpmac.h.

4.9.2.24 `int lspg_nextshot_struct::cy2_isnull`

Definition at line 320 of file pgpmac.h.

4.9.2.25 `int lspg_nextshot_struct::cy_isnull`

Definition at line 269 of file pgpmac.h.

4.9.2.26 `char* lspg_nextshot_struct::dsdir`

Directory for data relative to the ESAF home directory.

Definition at line 223 of file pgpmac.h.

4.9.2.27 `int lspg_nextshot_struct::dsdir_isnull`

Definition at line 224 of file pgpmac.h.

4.9.2.28 `double lspg_nextshot_struct::dsdist`

dataset defined detector distance

Definition at line 256 of file pgpmac.h.

4.9.2.29 `double lspg_nextshot_struct::dsdist2`

next image distance

Definition at line 310 of file pgpmac.h.

4.9.2.30 `int lspg_nextshot_struct::dsdist2_isnull`

Definition at line 311 of file pgpmac.h.

4.9.2.31 `int lspg_nextshot_struct::dsdist_isnull`

Definition at line 257 of file pgpmac.h.

4.9.2.32 double lspg_nextshot_struct::dsexp

dataset defined exposure time

Definition at line 235 of file pgpmac.h.

4.9.2.33 double lspg_nextshot_struct::dsexp2

next image exposure time

Definition at line 295 of file pgpmac.h.

4.9.2.34 int lspg_nextshot_struct::dsexp2_isnull

Definition at line 296 of file pgpmac.h.

4.9.2.35 int lspg_nextshot_struct::dsexp_isnull

Definition at line 236 of file pgpmac.h.

4.9.2.36 unsigned int lspg_nextshot_struct::dshpid

sample holder ID

Definition at line 262 of file pgpmac.h.

4.9.2.37 int lspg_nextshot_struct::dshpid_isnull

Definition at line 263 of file pgpmac.h.

4.9.2.38 double lspg_nextshot_struct::dskappa

dataset defined starting kappa angle

Definition at line 253 of file pgpmac.h.

4.9.2.39 double lspg_nextshot_struct::dskappa2

next image kappa position

Definition at line 307 of file pgpmac.h.

4.9.2.40 int lspg_nextshot_struct::dskappa2_isnull

Definition at line 308 of file pgpmac.h.

4.9.2.41 int lspg_nextshot_struct::dskappa_isnull

Definition at line 254 of file pgpmac.h.

4.9.2.42 `double lspg_nextshot_struct::dsnrg`

dataset defined energy

Definition at line 259 of file pgpmac.h.

4.9.2.43 `double lspg_nextshot_struct::dsnrg2`

next image energy

Definition at line 313 of file pgpmac.h.

4.9.2.44 `int lspg_nextshot_struct::dsnrg2_isnull`

Definition at line 314 of file pgpmac.h.

4.9.2.45 `int lspg_nextshot_struct::dsnrg_isnull`

Definition at line 260 of file pgpmac.h.

4.9.2.46 `double lspg_nextshot_struct::dsomega`

dataset defined starting omega angle

Definition at line 250 of file pgpmac.h.

4.9.2.47 `double lspg_nextshot_struct::dsomega2`

next image omega position

Definition at line 304 of file pgpmac.h.

4.9.2.48 `int lspg_nextshot_struct::dsomega2_isnull`

Definition at line 305 of file pgpmac.h.

4.9.2.49 `int lspg_nextshot_struct::dsomega_isnull`

Definition at line 251 of file pgpmac.h.

4.9.2.50 `char* lspg_nextshot_struct::dsoscaxis`

dataset defined oscillation axis (always omega)

Definition at line 232 of file pgpmac.h.

4.9.2.51 `char* lspg_nextshot_struct::dsoscaxis2`

next image oscillation axis (always "omega")

Definition at line 292 of file pgpmac.h.

4.9.2.52 `int lspg_nextshot_struct::dsoscaxis2_isnull`

Definition at line 293 of file pgpmac.h.

4.9.2.53 `int lspg_nextshot_struct::dsoscaxis_isnull`

Definition at line 233 of file pgpmac.h.

4.9.2.54 `double lspg_nextshot_struct::dsowidth`

dataset defined oscillation width

Definition at line 229 of file pgpmac.h.

4.9.2.55 `double lspg_nextshot_struct::dsowidth2`

next image oscillation width

Definition at line 289 of file pgpmac.h.

4.9.2.56 `int lspg_nextshot_struct::dsowidth2_isnull`

Definition at line 290 of file pgpmac.h.

4.9.2.57 `int lspg_nextshot_struct::dsowidth_isnull`

Definition at line 230 of file pgpmac.h.

4.9.2.58 `double lspg_nextshot_struct::dsphi`

dataset defined starting phi angle

Definition at line 247 of file pgpmac.h.

4.9.2.59 `double lspg_nextshot_struct::dsphi2`

next image phi position

Definition at line 301 of file pgpmac.h.

4.9.2.60 `int lspg_nextshot_struct::dsphi2_isnull`

Definition at line 302 of file pgpmac.h.

4.9.2.61 `int lspg_nextshot_struct::dsphi_isnull`

Definition at line 248 of file pgpmac.h.

4.9.2.62 `char* lspg_nextshot_struct::dspid`

ID string identifying this dataset.

Definition at line 226 of file pgpmac.h.

4.9.2.63 `int lspg_nextshot_struct::dspid_isnull`

Definition at line 227 of file pgpmac.h.

4.9.2.64 `pthread_mutex_t lspg_nextshot_struct::mutex`

Our mutex for sanity in the multi-threaded program.

Definition at line 218 of file pgpmac.h.

4.9.2.65 `int lspg_nextshot_struct::new_value_ready`

Our flag for the condition to wait for.

Definition at line 220 of file pgpmac.h.

4.9.2.66 `int lspg_nextshot_struct::no_rows_returned`

flag indicating that no rows were returned.

Definition at line 221 of file pgpmac.h.

4.9.2.67 `char* lspg_nextshot_struct::sfn`

file name

Definition at line 244 of file pgpmac.h.

4.9.2.68 `int lspg_nextshot_struct::sfn_isnull`

Definition at line 245 of file pgpmac.h.

4.9.2.69 `int lspg_nextshot_struct::sindex`

index of frame (used to generate the file extension)

Definition at line 283 of file pgpmac.h.

4.9.2.70 `int lspg_nextshot_struct::sindex2`

next image index number

Definition at line 334 of file pgpmac.h.

4.9.2.71 `int lspg_nextshot_struct::sindex2_isnull`

Definition at line 335 of file pgpmac.h.

4.9.2.72 `int lspg_nextshot_struct::sindex_isnull`

Definition at line 284 of file pgpmac.h.

4.9.2.73 long long lspg_nextshot_struct::skey

key identifying a particular image

Definition at line 238 of file pgpmac.h.

4.9.2.74 int lspg_nextshot_struct::skey_isnull

Definition at line 239 of file pgpmac.h.

4.9.2.75 double lspg_nextshot_struct::sstart

starting angle

Definition at line 241 of file pgpmac.h.

4.9.2.76 double lspg_nextshot_struct::sstart2

next image start angle

Definition at line 298 of file pgpmac.h.

4.9.2.77 int lspg_nextshot_struct::sstart2_isnull

Definition at line 299 of file pgpmac.h.

4.9.2.78 int lspg_nextshot_struct::sstart_isnull

Definition at line 242 of file pgpmac.h.

4.9.2.79 char* lspg_nextshot_struct::stype

"Normal" or "Gridsearch"

Definition at line 286 of file pgpmac.h.

4.9.2.80 char* lspg_nextshot_struct::stype2

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

Definition at line 337 of file pgpmac.h.

4.9.2.81 int lspg_nextshot_struct::stype2_isnull

Definition at line 338 of file pgpmac.h.

4.9.2.82 int lspg_nextshot_struct::stype_isnull

Definition at line 287 of file pgpmac.h.

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

- [pgpmac.h](#)

4.10 `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](#)

4.10.1 Detailed Description

Data collection running object.

Definition at line 923 of file `lspg.c`.

4.10.2 Field Documentation

4.10.2.1 `pthread_cond_t` `lspg_seq_run_prep_struct::cond`

Definition at line 925 of file `lspg.c`.

4.10.2.2 `pthread_mutex_t` `lspg_seq_run_prep_struct::mutex`

Definition at line 924 of file `lspg.c`.

4.10.2.3 `int` `lspg_seq_run_prep_struct::new_value_ready`

Definition at line 926 of file `lspg.c`.

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

- [lspg.c](#)

4.11 `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](#)

4.11.1 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 741 of file `lspg.c`.

4.11.2 Field Documentation

4.11.2.1 pthread_cond_t lspg_wait_for_detector_struct::cond

Definition at line 743 of file lspg.c.

4.11.2.2 pthread_mutex_t lspg_wait_for_detector_struct::mutex

Definition at line 742 of file lspg.c.

4.11.2.3 int lspg_wait_for_detector_struct::new_value_ready

Definition at line 744 of file lspg.c.

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

- [lspg.c](#)

4.12 lspgQueryQueueStruct Struct Reference

Store each query along with it's callback function.

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 [lspgQueryQueueStruct](#) *qq, PGresult *pgr)
Callback function for when a query returns a result.

4.12.1 Detailed Description

Store each query along with it's callback function.

All calls are asynchronous

Definition at line 31 of file kvredis.c.

4.12.2 Field Documentation

4.12.2.1 void(* lspgQueryQueueStruct::onResponse)(struct lspgQueryQueueStruct *qq, PGresult *pgr)

Callback function for when a query returns a result.

Definition at line 33 of file kvredis.c.

4.12.2.2 char lspgQueryQueueStruct::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](#)
- [lspg.c](#)

4.13 lspmac_bi_struct Struct Reference

Storage for binary inputs.

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

Data Fields

- int * [ptr](#)
points to the location in the status buffer
- pthread_mutex_t [mutex](#)
so we don't get confused
- int [mask](#)
mask for the bit in the status register
- int [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.

4.13.1 Detailed Description

Storage for binary inputs.

Definition at line 164 of file pgpmac.h.

4.13.2 Field Documentation

4.13.2.1 char* lspmac_bi_struct::changeEventOff

Event to send when the value changes to 0.

Definition at line 171 of file pgpmac.h.

4.13.2.2 char* lspmac_bi_struct::changeEventOn

Event to send when the value changes to 1.

Definition at line 170 of file pgpmac.h.

4.13.2.3 int lspmac_bi_struct::first_time

flag indicating we've not read the input even once

Definition at line 169 of file pgpmac.h.

4.13.2.4 `int lspmac_bi_struct::mask`

mask for the bit in the status register

Definition at line 167 of file `pgpmac.h`.

4.13.2.5 `pthread_mutex_t lspmac_bi_struct::mutex`

so we don't get confused

Definition at line 166 of file `pgpmac.h`.

4.13.2.6 `int lspmac_bi_struct::previous`

the previous value

Definition at line 168 of file `pgpmac.h`.

4.13.2.7 `int* lspmac_bi_struct::ptr`

points to the location in the status buffer

Definition at line 165 of file `pgpmac.h`.

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

- [pgpmac.h](#)

4.14 `lspmac_cmd_queue_struct` Struct Reference

PMAC command queue item.

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

Data Fields

- [pmac_cmd_t pcmd](#)
the pmac command to send
- `int no_reply`
1 = no reply is expected, 0 = expect a reply
- `struct timespec time_sent`
time this item was dequeued and sent to the pmac
- `unsigned char rbuff [1400]`
buffer for the returned bytes
- `void(* onResponse)(struct lspmac_cmd_queue_struct *, int, unsigned char *)`
function to call when response is received. args are (int fd, nreturned, buffer)

4.14.1 Detailed Description

PMAC command queue item.

Command queue items are fixed length to simplify memory management.

Definition at line 81 of file `pgpmac.h`.

4.14.2 Field Documentation

4.14.2.1 `int lspmac_cmd_queue_struct::no_reply`

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

Definition at line 83 of file `pgpmac.h`.

4.14.2.2 `void(* lspmac_cmd_queue_struct::onResponse)(struct lspmac_cmd_queue_struct *, int, unsigned char *)`

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

Definition at line 86 of file `pgpmac.h`.

4.14.2.3 `pmac_cmd_t lspmac_cmd_queue_struct::pcmd`

the pmac command to send

Definition at line 82 of file `pgpmac.h`.

4.14.2.4 `unsigned char lspmac_cmd_queue_struct::rbuff[1400]`

buffer for the returned bytes

Definition at line 85 of file `pgpmac.h`.

4.14.2.5 `struct timespec lspmac_cmd_queue_struct::time_sent`

time this item was dequeued and sent to the pmac

Definition at line 84 of file `pgpmac.h`.

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

- [pgpmac.h](#)

4.15 `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
- `int` [lspg_initialized](#)
bit flags: bit 0 = motor initialized by database, bit 1 = px.kvs value initialized
- `lskvs_kvs_list_t *` [presets](#)
list of preset positions

- `regex_t preset_regex`
buffer used by regex routines to find preset positions for this motor
- `void(* read)(struct lspmac_motor_struct *)`
method to read the motor status and position
- `int motion_seen`
set to 1 when motion has been verified to have started
- `struct lspmac_cmd_queue_struct * pq`
the queue item requesting motion. Used to check time request was made
- `char ** home`
pmac commands to home motor
- `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.
- `double update_resolution`
Change needs to be at least this big to report as a new position to the database.
- `char * update_format`
special format string to create text array for px.kvs update (lupdate)
- `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 statuss [64]`
short text summarizing status
- `int motor_num`
pmac motor number
- `int coord_num`
coordinate system this motor belongs to (0 if none)
- `char * axis`
the axis (X, Y, Z, etc) or null if not in a coordinate system
- `char * dac_mvar`
controlling mvariable as a string
- `char * name`
Name of motor as referred by ls database kvs table.
- `char * units`
string to use as the units
- `char * format`

- printf format*
- 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
- [lsredis_obj_t](#) * [u2c](#)
conversion from counts to units: 0.0 means not loaded yet
- double * [lut](#)
lookup table (instead of u2c)
- int [nlut](#)
length of lut
- double [max_speed](#)
our maximum speed (cts/msec)
- double [max_accel](#)
our maximum acceleration (cts/msec²)
- WINDOW * [win](#)
our ncurses window

4.15.1 Detailed Description

Motor information.

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

Definition at line 116 of file `pgpmac.h`.

4.15.2 Field Documentation

4.15.2.1 int `lspmac_motor_struct::actual_pos_cnts`

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

Definition at line 131 of file `pgpmac.h`.

4.15.2.2 int* `lspmac_motor_struct::actual_pos_cnts_p`

pointer to the `md2_status` structure to the actual position

Definition at line 130 of file `pgpmac.h`.

4.15.2.3 char* `lspmac_motor_struct::axis`

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

Definition at line 144 of file `pgpmac.h`.

4.15.2.4 pthread_cond_t `lspmac_motor_struct::cond`

used to signal when a motor is done moving

Definition at line 118 of file `pgpmac.h`.

4.15.2.5 int lspmac_motor_struct::coord_num

coordinate system this motor belongs to (0 if none)

Definition at line 143 of file pgpmac.h.

4.15.2.6 char* lspmac_motor_struct::dac_mvar

controlling mvariable as a string

Definition at line 145 of file pgpmac.h.

4.15.2.7 char* lspmac_motor_struct::format

printf format

Definition at line 148 of file pgpmac.h.

4.15.2.8 char lspmac_motor_struct::home**

pmac commands to home motor

Definition at line 127 of file pgpmac.h.

4.15.2.9 int lspmac_motor_struct::homing

Homing routine started.

Definition at line 128 of file pgpmac.h.

4.15.2.10 int lspmac_motor_struct::lspg_initialized

bit flags: bit 0 = motor initialized by database, bit 1 = px.kvs value initialized

Definition at line 120 of file pgpmac.h.

4.15.2.11 double* lspmac_motor_struct::lut

lookup table (instead of u2c)

Definition at line 154 of file pgpmac.h.

4.15.2.12 double lspmac_motor_struct::max_accel

our maximum acceleration (cts/msec²)

Definition at line 157 of file pgpmac.h.

4.15.2.13 double lspmac_motor_struct::max_speed

our maximum speed (cts/msec)

Definition at line 156 of file pgpmac.h.

4.15.2.14 `int lspmac_motor_struct::motion_seen`

set to 1 when motion has been verified to have started

Definition at line 124 of file pgpmac.h.

4.15.2.15 `int lspmac_motor_struct::motor_num`

pmac motor number

Definition at line 142 of file pgpmac.h.

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

function to move the motor

Definition at line 152 of file pgpmac.h.

4.15.2.17 `pthread_mutex_t lspmac_motor_struct::mutex`

coordinate waiting for motor to be done

Definition at line 117 of file pgpmac.h.

4.15.2.18 `char* lspmac_motor_struct::name`

Name of motor as referred by ls database kvs table.

Definition at line 146 of file pgpmac.h.

4.15.2.19 `int lspmac_motor_struct::nlut`

length of lut

Definition at line 155 of file pgpmac.h.

4.15.2.20 `int lspmac_motor_struct::not_done`

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

Definition at line 119 of file pgpmac.h.

4.15.2.21 `double lspmac_motor_struct::position`

scaled position

Definition at line 132 of file pgpmac.h.

4.15.2.22 `struct lspmac_cmd_queue_struct* lspmac_motor_struct::pq`

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

Definition at line 125 of file pgpmac.h.

4.15.2.23 `regex_t lspmac_motor_struct::preset_regex`

buffer used by regex routines to find preset positions for this motor

Definition at line 122 of file `pgpmac.h`.

4.15.2.24 `lskvs_kvs_list_t* lspmac_motor_struct::presets`

list of preset positions

Definition at line 121 of file `pgpmac.h`.

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

method to read the motor status and position

Definition at line 123 of file `pgpmac.h`.

4.15.2.26 `int lspmac_motor_struct::read_mask`

With `read_ptr` find bit to read for binary i/o.

Definition at line 151 of file `pgpmac.h`.

4.15.2.27 `int* lspmac_motor_struct::read_ptr`

With `read_mask` finds bit to read for binary i/o.

Definition at line 150 of file `pgpmac.h`.

4.15.2.28 `double lspmac_motor_struct::reported_position`

previous position reported to the database

Definition at line 133 of file `pgpmac.h`.

4.15.2.29 `int lspmac_motor_struct::requested_pos_cnts`

requested position

Definition at line 129 of file `pgpmac.h`.

4.15.2.30 `double lspmac_motor_struct::requested_position`

The position as requested by the user.

Definition at line 134 of file `pgpmac.h`.

4.15.2.31 `int lspmac_motor_struct::status1`

local copy of status1

Definition at line 138 of file `pgpmac.h`.

4.15.2.32 int* lspmac_motor_struct::status1_p

First 24 bit PMAC motor status word.

Definition at line 137 of file pgpmac.h.

4.15.2.33 int lspmac_motor_struct::status2

local copy of status2

Definition at line 140 of file pgpmac.h.

4.15.2.34 int* lspmac_motor_struct::status2_p

Sectond 24 bit PMAC motor status word.

Definition at line 139 of file pgpmac.h.

4.15.2.35 char lspmac_motor_struct::statuss[64]

short text summarizing status

Definition at line 141 of file pgpmac.h.

4.15.2.36 lsredis_obj_t* lspmac_motor_struct::u2c

conversion from counts to units: 0.0 means not loaded yet

Definition at line 153 of file pgpmac.h.

4.15.2.37 char* lspmac_motor_struct::units

string to use as the units

Definition at line 147 of file pgpmac.h.

4.15.2.38 char* lspmac_motor_struct::update_format

special format string to create text array for px.kvs update (lsupdate)

Definition at line 136 of file pgpmac.h.

4.15.2.39 double 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 135 of file pgpmac.h.

4.15.2.40 WINDOW* lspmac_motor_struct::win

our ncurses window

Definition at line 158 of file pgpmac.h.

4.15.2.41 char* lspmac_motor_struct::write_fmt

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

Definition at line 149 of file pgpmac.h.

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

- [pgpmac.h](#)

4.16 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
- char *(* [getstr](#))(struct Isredis_obj_struct *)
return a string representation of this object
- double(* [getd](#))(struct Isredis_obj_struct *)
return a double floating point version
- long int(* [getl](#))(struct Isredis_obj_struct *)
return a long value
- int [hits](#)
number of times we've searched for this key

4.16.1 Detailed Description

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

Definition at line 35 of file pgpmac.h.

4.16.2 Field Documentation

4.16.2.1 `pthread_cond_t lsredis_obj_struct::cond`

wait for a valid value

Definition at line 37 of file pgpmac.h.

4.16.2.2 `char* lsredis_obj_struct::events_name`

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

Definition at line 42 of file pgpmac.h.

4.16.2.3 `double(* lsredis_obj_struct::getd)(struct lsredis_obj_struct *)`

return a double floating point version

Definition at line 46 of file pgpmac.h.

4.16.2.4 `long int(* lsredis_obj_struct::getl)(struct lsredis_obj_struct *)`

return a long value

Definition at line 47 of file pgpmac.h.

4.16.2.5 `char*(* lsredis_obj_struct::getstr)(struct lsredis_obj_struct *)`

return a string representation of this object

Definition at line 45 of file pgpmac.h.

4.16.2.6 `int lsredis_obj_struct::hits`

number of times we've searched for this key

Definition at line 48 of file pgpmac.h.

4.16.2.7 `char* lsredis_obj_struct::key`

The redis key for this object.

Definition at line 41 of file pgpmac.h.

4.16.2.8 `pthread_mutex_t lsredis_obj_struct::mutex`

Don't let anyone use an old value.

Definition at line 36 of file pgpmac.h.

4.16.2.9 `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 38 of file pgpmac.h.

4.16.2.10 `char lsredis_obj_struct::valid`

1 if we think the value is good, 0 otherwise

Definition at line 39 of file `pgpmac.h`.

4.16.2.11 `char* lsredis_obj_struct::value`

our value

Definition at line 44 of file `pgpmac.h`.

4.16.2.12 `int lsredis_obj_struct::value_length`

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

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

4.16.2.13 `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 40 of file `pgpmac.h`.

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

- [pgpmac.h](#)

4.17 `lstimer_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
- `unsigned long int next_secs`
epoch (seconds) of next alarm
- `unsigned long int next_nsecs`
nano seconds of next alarm
- `unsigned long int delay_secs`
number of seconds for a periodic delay
- `unsigned long int delay_nsecs`
nano seconds of delay
- `unsigned long int last_secs`
the last time this timer was triggered
- `unsigned long int last_nsecs`
the last time this timer was triggered
- `unsigned long int init_secs`

- our initialization time*
 • unsigned long int `init_nsecs`
our initialization time

4.17.1 Detailed Description

Everything we need to know about a timer.

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

4.17.2 Field Documentation

4.17.2.1 unsigned long int `lstimer_list_struct::delay_nsecs`

nano seconds of delay

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

4.17.2.2 unsigned long int `lstimer_list_struct::delay_secs`

number of seconds for a periodic delay

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

4.17.2.3 char `lstimer_list_struct::event[LSEVENTS_EVENT_LENGTH]`

the event to send

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

4.17.2.4 unsigned long int `lstimer_list_struct::init_nsecs`

our initialization time

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

4.17.2.5 unsigned long int `lstimer_list_struct::init_secs`

our initialization time

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

4.17.2.6 unsigned long int `lstimer_list_struct::last_nsecs`

the last time this timer was triggered

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

4.17.2.7 unsigned long int `lstimer_list_struct::last_secs`

the last time this timer was triggered

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

4.17.2.8 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.

4.17.2.9 unsigned long int ltimer_list_struct::next_nsecs

nano seconds of next alarm

Definition at line 27 of file ltimer.c.

4.17.2.10 unsigned long int ltimer_list_struct::next_secs

epoch (seconds) of next alarm

Definition at line 26 of file ltimer.c.

4.17.2.11 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](#)

4.18 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](#)

4.18.1 Detailed Description

The block of memory retrieved in a status request.

Definition at line 201 of file `lspmac.c`.

4.18.2 Field Documentation

4.18.2.1 int md2StatusStruct::acc11c_1

Definition at line 268 of file lspmacc.c.

4.18.2.2 int md2StatusStruct::acc11c_2

Definition at line 269 of file lspmacc.c.

4.18.2.3 int md2StatusStruct::acc11c_3

Definition at line 270 of file lspmacc.c.

4.18.2.4 int md2StatusStruct::acc11c_5

Definition at line 271 of file lspmacc.c.

4.18.2.5 int md2StatusStruct::acc11c_6

Definition at line 272 of file lspmacc.c.

4.18.2.6 int md2StatusStruct::alignx_act_pos

Definition at line 252 of file lspmacc.c.

4.18.2.7 int md2StatusStruct::alignx_status_1

Definition at line 218 of file lspmacc.c.

4.18.2.8 int md2StatusStruct::alignx_status_2

Definition at line 235 of file lspmacc.c.

4.18.2.9 int md2StatusStruct::aligny_act_pos

Definition at line 253 of file lspmacc.c.

4.18.2.10 int md2StatusStruct::aligny_status_1

Definition at line 219 of file lspmacc.c.

4.18.2.11 int md2StatusStruct::aligny_status_2

Definition at line 236 of file lspmacc.c.

4.18.2.12 int md2StatusStruct::alignz_act_pos

Definition at line 254 of file lspmacc.c.

4.18.2.13 `int md2StatusStruct::alignz_status_1`

Definition at line 220 of file `lspmac.c`.

4.18.2.14 `int md2StatusStruct::alignz_status_2`

Definition at line 237 of file `lspmac.c`.

4.18.2.15 `int md2StatusStruct::analyzer_act_pos`

Definition at line 255 of file `lspmac.c`.

4.18.2.16 `int md2StatusStruct::analyzer_status_1`

Definition at line 221 of file `lspmac.c`.

4.18.2.17 `int md2StatusStruct::analyzer_status_2`

Definition at line 238 of file `lspmac.c`.

4.18.2.18 `int md2StatusStruct::aperturey_act_pos`

Definition at line 257 of file `lspmac.c`.

4.18.2.19 `int md2StatusStruct::aperturey_status_1`

Definition at line 223 of file `lspmac.c`.

4.18.2.20 `int md2StatusStruct::aperturey_status_2`

Definition at line 240 of file `lspmac.c`.

4.18.2.21 `int md2StatusStruct::aperturez_act_pos`

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

4.18.2.22 `int md2StatusStruct::aperturez_status_1`

Definition at line 224 of file `lspmac.c`.

4.18.2.23 `int md2StatusStruct::aperturez_status_2`

Definition at line 241 of file `lspmac.c`.

4.18.2.24 `int md2StatusStruct::back_dac`

Definition at line 274 of file `lspmac.c`.

4.18.2.25 int md2StatusStruct::copy_act_pos

Definition at line 259 of file lspmac.c.

4.18.2.26 int md2StatusStruct::copy_status_1

Definition at line 225 of file lspmac.c.

4.18.2.27 int md2StatusStruct::copy_status_2

Definition at line 242 of file lspmac.c.

4.18.2.28 int md2StatusStruct::capz_act_pos

Definition at line 260 of file lspmac.c.

4.18.2.29 int md2StatusStruct::capz_status_1

Definition at line 226 of file lspmac.c.

4.18.2.30 int md2StatusStruct::capz_status_2

Definition at line 243 of file lspmac.c.

4.18.2.31 int md2StatusStruct::centerx_act_pos

Definition at line 262 of file lspmac.c.

4.18.2.32 int md2StatusStruct::centerx_status_1

Definition at line 228 of file lspmac.c.

4.18.2.33 int md2StatusStruct::centerx_status_2

Definition at line 245 of file lspmac.c.

4.18.2.34 int md2StatusStruct::centery_act_pos

Definition at line 263 of file lspmac.c.

4.18.2.35 int md2StatusStruct::centery_status_1

Definition at line 229 of file lspmac.c.

4.18.2.36 int md2StatusStruct::centery_status_2

Definition at line 246 of file lspmac.c.

4.18.2.37 int md2StatusStruct::dummy1

Definition at line 216 of file lspmac.c.

4.18.2.38 int md2StatusStruct::dummy2

Definition at line 233 of file lspmac.c.

4.18.2.39 int md2StatusStruct::dummy3

Definition at line 250 of file lspmac.c.

4.18.2.40 int md2StatusStruct::dummy4

Definition at line 277 of file lspmac.c.

4.18.2.41 int md2StatusStruct::dummy5

Definition at line 278 of file lspmac.c.

4.18.2.42 int md2StatusStruct::dummy6

Definition at line 279 of file lspmac.c.

4.18.2.43 int md2StatusStruct::dummy7

Definition at line 280 of file lspmac.c.

4.18.2.44 int md2StatusStruct::dummy8

Definition at line 281 of file lspmac.c.

4.18.2.45 int md2StatusStruct::dummy9

Definition at line 282 of file lspmac.c.

4.18.2.46 int md2StatusStruct::dummyA

Definition at line 283 of file lspmac.c.

4.18.2.47 int md2StatusStruct::dummyB

Definition at line 284 of file lspmac.c.

4.18.2.48 int md2StatusStruct::front_dac

Definition at line 273 of file lspmac.c.

4.18.2.49 int md2StatusStruct::fs_has_opened

Definition at line 288 of file lspmacc.c.

4.18.2.50 int md2StatusStruct::fs_has_opened_globally

Definition at line 289 of file lspmacc.c.

4.18.2.51 int md2StatusStruct::fs_is_open

Definition at line 286 of file lspmacc.c.

4.18.2.52 int md2StatusStruct::kappa_act_pos

Definition at line 264 of file lspmacc.c.

4.18.2.53 int md2StatusStruct::kappa_status_1

Definition at line 230 of file lspmacc.c.

4.18.2.54 int md2StatusStruct::kappa_status_2

Definition at line 247 of file lspmacc.c.

4.18.2.55 int md2StatusStruct::moving_flags

Definition at line 292 of file lspmacc.c.

4.18.2.56 int md2StatusStruct::number_passes

Definition at line 290 of file lspmacc.c.

4.18.2.57 int md2StatusStruct::omega_act_pos

Definition at line 251 of file lspmacc.c.

4.18.2.58 int md2StatusStruct::omega_status_1

Definition at line 217 of file lspmacc.c.

4.18.2.59 int md2StatusStruct::omega_status_2

Definition at line 234 of file lspmacc.c.

4.18.2.60 int md2StatusStruct::phi_act_pos

Definition at line 265 of file lspmacc.c.

4.18.2.61 int md2StatusStruct::phi_status_1

Definition at line 231 of file lspmac.c.

4.18.2.62 int md2StatusStruct::phi_status_2

Definition at line 248 of file lspmac.c.

4.18.2.63 int md2StatusStruct::phiscan

Definition at line 287 of file lspmac.c.

4.18.2.64 int md2StatusStruct::scint_act_pos

Definition at line 261 of file lspmac.c.

4.18.2.65 int md2StatusStruct::scint_piezo

Definition at line 275 of file lspmac.c.

4.18.2.66 int md2StatusStruct::scint_status_1

Definition at line 227 of file lspmac.c.

4.18.2.67 int md2StatusStruct::scint_status_2

Definition at line 244 of file lspmac.c.

4.18.2.68 int md2StatusStruct::zoom_act_pos

Definition at line 256 of file lspmac.c.

4.18.2.69 int md2StatusStruct::zoom_status_1

Definition at line 222 of file lspmac.c.

4.18.2.70 int md2StatusStruct::zoom_status_2

Definition at line 239 of file lspmac.c.

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

- [lspmac.c](#)

4.19 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.

4.19.1 Detailed Description

PMAC ethernet packet definition.

Taken directly from the Delta Tau documentation.

Definition at line 68 of file pgpmac.h.

4.19.2 Field Documentation

4.19.2.1 unsigned char tagEthernetCmd::bData[1492]

The data buffer, if required.

Definition at line 74 of file pgpmac.h.

4.19.2.2 unsigned char tagEthernetCmd::Request

The command to run (VR_PMAC_GETMEM, etc).

Definition at line 70 of file pgpmac.h.

4.19.2.3 unsigned char tagEthernetCmd::RequestType

VR_UPLOAD or VR_DOWNLOAD.

Definition at line 69 of file pgpmac.h.

4.19.2.4 unsigned short tagEthernetCmd::wIndex

Command parameter 2.

Definition at line 72 of file pgpmac.h.

4.19.2.5 unsigned short tagEthernetCmd::wLength

Number of bytes in bData.

Definition at line 73 of file pgpmac.h.

4.19.2.6 unsigned short tagEthernetCmd::wValue

Command parameter 1.

Definition at line 71 of file pgpmac.h.

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

- [pgpmac.h](#)

Chapter 5

File Documentation

5.1 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

5.1.1 Macro Definition Documentation

5.1.1.1 `#define LS_PG_QUERY_QUEUE_LENGTH 512`

Definition at line 12 of file kvredis.c.

5.1.1.2 `#define LS_PG_QUERY_STRING_LENGTH 512`

Definition at line 13 of file kvredis.c.

5.1.1.3 `#define LS_PG_STATE_IDLE 1`

Definition at line 19 of file kvredis.c.

5.1.1.4 `#define LS_PG_STATE_INIT -4`

Definition at line 15 of file kvredis.c.

5.1.1.5 `#define LS_PG_STATE_INIT_POLL -3`

Definition at line 16 of file kvredis.c.

5.1.1.6 `#define LS_PG_STATE_RECV 4`

Definition at line 22 of file kvredis.c.

5.1.1.7 `#define LS_PG_STATE_RESET -2`

Definition at line 17 of file kvredis.c.

5.1.1.8 `#define LS_PG_STATE_RESET_POLL -1`

Definition at line 18 of file kvredis.c.

5.1.1.9 #define LS_PG_STATE_SEND 2

Definition at line 20 of file kvredis.c.

5.1.1.10 #define LS_PG_STATE_SEND_FLUSH 3

Definition at line 21 of file kvredis.c.

5.1.2 Typedef Documentation

5.1.2.1 typedef struct lspgQueryQueueStruct lspg_query_queue_t

Store each query along with it's callback function.

All calls are asynchronous

5.1.3 Function Documentation

5.1.3.1 void addRead (void * *data*)

Definition at line 108 of file kvredis.c.

```

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

5.1.3.2 void addWrite (void * *data*)

Definition at line 118 of file kvredis.c.

```

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

5.1.3.3 void cleanup (void * *data*)

Definition at line 128 of file kvredis.c.

```

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

5.1.3.4 void debugCB (redisAsyncContext * *ac*, void * *reply*, void * *privdata*)

Definition at line 63 of file kvredis.c.

```

{
    redisReply *r;
    int i;

    r = (redisReply *)reply;
```

```

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

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

case REDIS_REPLY_ERROR:
    printf( "ERROR: %s\n", r->str);
    break;

case REDIS_REPLY_INTEGER:
    printf( "Integer: %lld\n", r->integer);
    break;

case REDIS_REPLY_NIL:
    printf( "(nil)\n");
    break;

case REDIS_REPLY_STRING:
    printf( "STRING: %s\n", r->str);
    break;

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

default:
    printf( "Unknown type %d\n", r->type);
}
}

```

5.1.3.5 void delRead (void * data)

Definition at line 113 of file kvredis.c.

```

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

```

5.1.3.6 void delWrite (void * data)

Definition at line 123 of file kvredis.c.

```

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

```

5.1.3.7 void fd_service (struct pollfd * evt)

Definition at line 634 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);
}

```

5.1.3.8 void lspg_allkvs_cb (lspg_query_queue_t * qqp, PGresult * pgr)

Definition at line 134 of file kvredis.c.

```

{
    int kvname_col, kvvalue_col, kvseq_col, kvdbrtype_col;
    int i;
    int need_quotes;
    int seq;

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

        need_quotes = strchr( PQgetvalue( pgr, i, kvvalue_col), ' ') == NULL ? 0 :
            1;
        redisAsyncCommand( cmdac, debugCB, NULL, "HMSET %s VALUE %s%s%s
            SEQ %d DBRTYPE %d",
            PQgetvalue( pgr, i, kvname_col),
            need_quotes ? "\"0x42" : "",
            PQgetvalue( pgr, i, kvvalue_col),
            need_quotes ? "\"0x42" : "",
            atoi(PQgetvalue( pgr, i, kvseq_col)),
            atoi(PQgetvalue( pgr, i, kvdbrtype_col))
        );
        if( need_quotes)
            fprintf( stderr, "lspg_allkvs_cb: %s %s\n", PQgetvalue( pgr, i,
                kvname_col), PQgetvalue( pgr, i, kvvalue_col));
    }

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

    for( i=0; i<PQntuples( pgr); i++) {
        redisAsyncCommand( cmdac, debugCB, NULL, "PUBLISH
            REDIS_KV_CONNECTOR '%s %s'", PQgetvalue( pgr, i, kvname_col), PQgetvalue( pgr, i, kvseq_col))
        ;
    }
}

```

5.1.3.9 void lspg_flush ()

Flush psql output buffer (ie, send the query)

Definition at line 410 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;
}
}

```

5.1.3.10 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 442 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;
}
}

```

5.1.3.11 PQnoticeProcessor lspg_notice_processor (void * arg, const char * msg)

Definition at line 180 of file kvredis.c.

```

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

```

5.1.3.12 void lspg_pg_connect ()

Connect to the pg server.

Definition at line 323 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;
}
}

```

5.1.3.13 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 541 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();
    }
}
}
}

```

5.1.3.14 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 189 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;
}

```

5.1.3.15 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 232 of file kvredis.c.

```

    {
        int idx;
        va_list arg_ptr;

        //
        // Pause the thread while we service the queue
        //
        if( lspg_query_queue_on + 1 == lspg_query_queue_off
        ) {
            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++;
    };

```

5.1.3.16 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 209 of file kvredis.c.

```

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

```

5.1.3.17 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 219 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;
}

```

5.1.3.18 void lspg_receive ()

Receive a result of a query.

Definition at line 264 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->q, 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);
    }
}
}

```

5.1.3.19 void lspg_send_next_query ()

send the next queued query to the DB server

Definition at line 494 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;
}
}
}
}

```

5.1.3.20 main ()

Definition at line 653 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, debugCB, 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.
        // Periodically try again. Should possibly arrange to reconnect
        // to signalfd but that's unlikely to be necessary.
        //
        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;
    }
}

fprintf( stderr, "nfda: %d\n", nfda);

pollrtn = poll( fda, nfda, poll_timeout_ms);

```

```

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

```

5.1.3.21 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);
}

```

5.1.4 Variable Documentation

5.1.4.1 redisAsyncContext * cmdac [static]

Definition at line 9 of file kvredis.c.

5.1.4.2 struct pollfd cmdfd [static]

poll info for redis command channel

Definition at line 50 of file kvredis.c.

5.1.4.3 int kvseq = 0 [static]

used to synchronize pg.kvs and redis

Definition at line 26 of file kvredis.c.

5.1.4.4 int ls_pg_state = LS_PG_STATE_INIT [static]

State of the lspg state machine.

Definition at line 24 of file kvredis.c.

5.1.4.5 PostgresPollingStatusType lspg_connectPoll_response [static]

Used to determine state while connecting.

Definition at line 46 of file kvredis.c.

5.1.4.6 lspg_query_queue_t lspg_query_queue[LS_PG_QUERY_QUEUE_LENGTH] [static]

Our query queue.

Definition at line 37 of file kvredis.c.

5.1.4.7 `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.

5.1.4.8 `unsigned int lspg_query_queue_on = 0` `[static]`

Next position to add something to the queue.

Definition at line 38 of file kvredis.c.

5.1.4.9 `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.

5.1.4.10 `PostgresPollingStatusType lspg_resetPoll_response` `[static]`

Used to determine state while reconnecting.

Definition at line 47 of file kvredis.c.

5.1.4.11 `struct pollfd lspgfd` `[static]`

our poll info

Definition at line 48 of file kvredis.c.

5.1.4.12 `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.

5.1.4.13 `PGconn* q = NULL` `[static]`

Database connector.

Definition at line 45 of file kvredis.c.

5.1.4.14 `redisAsyncContext* subac` `[static]`

Definition at line 9 of file kvredis.c.

5.1.4.15 `struct pollfd subfd` `[static]`

poll info for redis subscribe channel

Definition at line 49 of file kvredis.c.

5.2 lsevents.c File Reference

event subsystem for inter-pgpmac communication

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

Data Structures

- struct [lsevents_queue_struct](#)
Storage definition for the events.
- struct [lsevents_listener_struct](#)
Linked list of event listeners.

Macros

- #define [LSEVENTS_QUEUE_LENGTH](#) 2096

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

5.2.1 Detailed Description

event subsystem for inter-pgpmac communication

Date

2012

Author

Keith Brister

Copyright

All Rights Reserved

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

5.2.2 Macro Definition Documentation

5.2.2.1 #define LSEVENTS_QUEUE_LENGTH 2096

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

5.2.3 Typedef Documentation

5.2.3.1 typedef struct lsevents_listener_struct lsevents_listener_t

Linked list of event listeners.

5.2.3.2 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.

5.2.4 Function Documentation

5.2.4.1 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 77 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);
}

```

5.2.4.2 void lsevents_init()

Initialize this module.

Definition at line 206 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);
}

```

5.2.4.3 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 122 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);
}
}

```

5.2.4.4 void lsevents_run ()

Start up the thread and get out of the way.

Definition at line 214 of file lsevents.c.

```

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

```

5.2.4.5 void lsevents_send_event (char *fmt, ...)

Call the callback routines for the given event.

Parameters

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

Definition at line 45 of file lsevents.c.

```

{

char event[LSEVENTS_EVENT_LENGTH];
char *sp;
va_list arg_ptr;

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

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

pthread_mutex_lock( &lsevents_queue_mutex);

```

```

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

sp = lsevents_queue[(lsevents_queue_on++) %
    LSEVENTS_QUEUE_LENGTH].event;
strncpy( sp, event, LSEVENTS_EVENT_LENGTH);
sp[LSEVENTS_EVENT_LENGTH - 1] = 0;

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

```

5.2.4.6 void* lsevents_worker (void * dummy)

Our worker.

Parameters

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

Definition at line 157 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]);
    event = strdup( ep->event, LSEVENTS_EVENT_LENGTH
        -1);
    event[LSEVENTS_EVENT_LENGTH-1] = 0;

    //
    // 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
    //
    pthread_mutex_lock( &lsevents_listener_mutex);
    for( p = lsevents_listeners_p; p != NULL; p = p->next
    ) {
        if( regexec( &p->re, event, 0, NULL, 0) == 0 ) {
            p->cb( event);
        }
    }
    free( event);

    pthread_mutex_unlock( &lsevents_listener_mutex);
}
return NULL;
}

```

5.2.5 Variable Documentation

5.2.5.1 `pthread_mutex_t lsevents_listener_mutex` `[static]`

mutex to protect the listener linked list

Definition at line 37 of file lsevents.c.

5.2.5.2 `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.

5.2.5.3 `lsevents_queue_t lsevents_queue[LSEVENTS_QUEUE_LENGTH]` `[static]`

simple list of events

Definition at line 21 of file lsevents.c.

5.2.5.4 `pthread_cond_t lsevents_queue_cond` `[static]`

condition to pause the queue if needed

Definition at line 39 of file lsevents.c.

5.2.5.5 `pthread_mutex_t lsevents_queue_mutex` `[static]`

mutex to protect the event queue

Definition at line 38 of file lsevents.c.

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

next queue location to read

Definition at line 23 of file lsevents.c.

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

next queue location to write

Definition at line 22 of file lsevents.c.

5.2.5.8 `pthread_t lsevents_thread` `[static]`

thread to run the event queue

Definition at line 36 of file lsevents.c.

5.3 Iskvs.c File Reference

Support for the remote access client key value pairs.

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

Functions

- double [lskvs_find_preset_position](#) ([lspmac_motor_t](#) *mp, char *name, int *err)
find a postion for a given preset name
- void [lskvs_regcomp](#) (regex_t *preg, int cflags, char *fmt,...)
Utility wrapper for regcomp providing printf style formatting.
- void [lskvs_set](#) (char *k, char *v)
Set the value of a kv pair Create the pair if the key does not exist.
- [lskvs_kvs_t](#) * [lskvs_get](#) (char *k)
Find the kv pair object Return with a pointer to the structure or NULL if not found.
- void [lskvs_init](#) ()
Initialize lskvs objects.
- void [lskvs_run](#) ()
Run things.

Variables

- [lskvs_kvs_t](#) * [lskvs_kvs](#) = NULL
our list (or at least the start of it
- pthread_rwlock_t [lskvs_rwlock](#)
needed to protect the list

5.3.1 Detailed Description

Support for the remote access client key value pairs.

Date

2012

Author

Keith Brister

Copyright

All Rights Reserved

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

5.3.2 Function Documentation

5.3.2.1 double lskvs_find_preset_position (lspmact_motor_t * mp, char * name, int * err)

find a postion for a given preset name

Parameters

<i>mp</i>	Motor pointer
<i>name</i>	The preset to search for
<i>err</i>	set to non-zero on error, ignored if null

Definition at line 21 of file lskvs.c.

```

regmatch_t pmatch[4], qmatch[4];
double rtn;
lskvs_kvs_list_t
    *position_kv = NULL,
    *name_kv      = NULL;
int e;

*err = -4;
if( name == NULL || *name == 0)
    return 0.0;

*err = 0;
for( name_kv = mp->presets; name_kv != NULL; name_kv = name_kv->next
    ) {
    if( strcmp( name, name_kv->kvs->v) == 0) {
        //
        // We found the correct preset, now get the index
        //
        e = regexec( &(amp;mp->preset_regex), name_kv->kvs->k, 4, pmatch,
            0);
        if( e != 0) {
            lslogging_log_message( "
lskvs_find_preset_position: could not parse name key '%s'", name_kv->kvs->k);
            if( err != NULL)
                *err = e;
            return 0.0;
        }

        for( position_kv = mp->presets; position_kv != NULL; position_kv =
            position_kv->next) {
            if( position_kv == name_kv)
                continue;

            e = regexec( &(amp;mp->preset_regex), position_kv->kvs->k,
                4, qmatch, 0);
            if( e != 0) {
                lslogging_log_message( "
lskvs_find_preset_position: could not parse position key '%s'", position_kv->kvs->k);
                if( err != NULL)
                    *err = e;
                return 0.0;
            }

            if( strncmp( name_kv->kvs->k, position_kv->kvs->k, qmatch[2].rm_eo
                + 1) == 0) {
                break;
            }
        }
        if( position_kv != NULL)
            break;
    }
}

if( name_kv != NULL || position_kv != NULL) {
    errno = 0;
    rtn = strtod( position_kv->kvs->v, NULL);
    if( errno != 0) {
        lslogging_log_message( "lskvs_find_preset_position:
        bad preset value for motor %s, preset %s, value '%s'", mp->name, name,
        position_kv->kvs->v);
        if( err != NULL)
            *err = -2;
        return 0.0;
    }
    return rtn;
}
lslogging_log_message( "lskvs_find_preset_position:
could not find preset for motor %s, preset %s", mp->name, name);
if( err != NULL)
    *err = -3;
return 0.0;
}

```

5.3.2.2 lskvs_kvs_t* lskvs_get(char * k)

Find the kv pair object Return with a pointer to the structure or NULL if not found.

Parameters

in	k	key name to search for
----	---	------------------------

Definition at line 252 of file Iskvs.c.

```

    {
lskvs_kvs_t
    *rtn;

pthread_rwlock_rdlock( &lskvs_rwlock);
rtn = lskvs_kvs;
pthread_rwlock_unlock( &lskvs_rwlock);

while(rtn != NULL) {
    if( strcmp( rtn->k, k) == 0)
        break;
    rtn = rtn->next;
}
return rtn;
}

```

5.3.2.3 void Iskvs_init ()

Initialize Iskvs objects.

Definition at line 273 of file Iskvs.c.

```

{
pthread_rwlock_init( &lskvs_rwlock, NULL);
}

```

5.3.2.4 void Iskvs_regcomp (regex_t * preg, int cflags, char * fmt, ...)

Utility wrapper for regcomp providing printf style formatting.

Parameters

<i>preg</i>	Buffer for the compile regex object
<i>cflags</i>	See regcomp man page
<i>fmt</i>	Printf style formatting string
...	Argument list specified by fmt

< no reason our search strings should ever be this big

Definition at line 92 of file Iskvs.c.

```

{
struct regerror_struct {
    int errcode;
    char *errstr;
};
static struct regerror_struct regerrors[] = {
    { REG_BADBR, "Invalid use of back reference operator."},
    { REG_BADPAT, "Invalid use of pattern operators such as group or list."},
    { REG_BADRPT, "Invalid use of repetition operators such as using '*' as
        the first character."},
    { REG_EBRACE, "Un-matched brace interval operators."},
    { REG_EBRACK, "Un-matched bracket list operators."},
    { REG_ECOLLATE, "Invalid collating element."},
    { REG_ECTYPE, "Unknown character class name."},
    { REG_EEND, "Non specific error. This is not defined by POSIX.2."},
    { REG_EESCAPE, "Trailing backslash."},
    { REG_EPAREN, "Un-matched parenthesis group operators."},
    { REG_ERANGE, "Invalid use of the range operator, e.g., the ending point
        of the range occurs prior to the starting point."},
    { REG_ESIZE, "Compiled regular expression requires a pattern buffer
        larger than 64Kb. This is not defined by POSIX.2."},
    { REG_ESPACE, "The regex routines ran out of memory."},
    { REG_ESUBREG, "Invalid back reference to a subexpression."},
    { 0, "No errors"}
};
}

```

```

va_list arg_ptr;
char s[512];
int err;

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

err = regcomp( preg, s, cflags);
if( err != 0) {
    int i;

    for( i=0; regerrors[i].errcode != 0; i++)
        if( regerrors[i].errcode == err)
            break;

    if( regerrors[i].errcode != 0) {
        lslogging_log_message( "lskvs_regcomp: could not
        compile regular experssion '%s'", s);
        lslogging_log_message( "lskvs_regcomp: regcomp
        returned %d: %s", err, regerrors[i]);
    }
}
}

```

5.3.2.5 void lskvs_run ()

Run things.

Really, there is nothing to run. There is no need for a worker thread here but this has been added so we can add lskvs just like any other module to the pgpmac project. Maybe one day we'll need to add a thread and this little routine can be celebrated as being far sighted, ahead of its time.

Definition at line 283 of file lskvs.c.

```

{
}

```

5.3.2.6 void lskvs_set(char * k, char * v)

Set the value of a kv pair Create the pair if the key does not exist.

If more than one thread tries to create the same key at the same time it is possible for the list to contain multiple versions. Not good. But also not possible if only one thread has the job of create the pairs in the first place. Alternatively just grab the write lock at the beginning and hold it until the end. The advantage of having only one thread calling lskvs_set is that it wont slow down the other threads that just want to read things. In any case, we'll likely never see so much action for any of this to make a differene.

Parameters

<i>k</i>	The name of the key
<i>v</i>	The value to assign to the key

Definition at line 156 of file lskvs.c.

```

{
    lskvs_kvs_t
    *root,
    *p;

    lslogging_log_message( "lskvs_set: k: '%s', v: '%s'", k
    , v);

    // Don't bother with empty keys
    //
    if( k == NULL || *k == 0)
        return;

    pthread_rwlock_rdlock( &lskvs_rwlock);
}

```

```

root = lskvs_kvs;
pthread_rwlock_unlock( &lskvs_rwlock);

for( p=root; p != NULL; p = p->next) {
    if( strcmp( p->k, k) == 0) {
        break;
    }
}

if( p == NULL) {
    //
    // Add a new list item
    //
    p = calloc( 1, sizeof( *p));
    if( p == NULL) {
        lslogging_log_message( "lskvs_set: out of memory for
            kv struct (%d bytes", sizeof( *p));
        exit( -1);
    }

    p->k = calloc( strlen(k)+1, sizeof( *k));
    if( p->k == NULL) {
        lslogging_log_message( "lskvs_set: out of memory for
            k (%d bytes)", strlen( k)+1);
        exit( -1);
    }
    strcpy( p->k, k);
    p->k[strlen(k)] = 0;

    // leave a little room to grow
    //
    if( v == NULL || *v == 0)
        p->v1 = 32;
    else
        p->v1 = strlen(v) + 32;

    p->v = calloc( p->v1, sizeof( *v));
    if( p->v == NULL) {
        lslogging_log_message( "lskvs_set: out of memory for
            v (%d bytes)", p->v1);
        exit( -1);
    }

    if( v == NULL || *v == 0)
        *(p->v) = 0;
    else
        strcpy( p->v, v);

    p->v[p->v1-1] = 0;

    pthread_rwlock_init( &p->l, NULL);

    pthread_rwlock_wrlock( &lskvs_rwlock);
    p->next = lskvs_kvs;
    lskvs_kvs = p;
    pthread_rwlock_unlock( &lskvs_rwlock);

    lsevents_send_event( "NewKV");
} else {
    //
    // Just update the value
    // Assume the database only sent us an update because
    // the old and new values are different
    //
    pthread_rwlock_wrlock( &(p->l));
    if( strlen( v) > p->v1-1) {
        free( p->v);

        p->v1 = strlen(v) + 32;
        p->v = calloc( p->v1, 1);
        if( p->v == NULL) {
            lslogging_log_message( "lskvs_set: out of memory
                for re-calloc of v (%d bytes)", p->v1);
            exit( -1);
        }
    }
    strcpy( p->v, v);
    p->v[p->v1-1] = 0;
    pthread_rwlock_unlock( &(p->l));
}
}

```

5.3.3 Variable Documentation

5.3.3.1 `lskvs_kvs_t*` `lskvs_kvs` = NULL

our list (or at least the start of it

Definition at line 11 of file `lskvs.c`.

5.3.3.2 `pthread_rwlock_t` `lskvs_rwlock`

needed to protect the list

Definition at line 12 of file `lskvs.c`.

5.4 `lslogging.c` File Reference

Logs messages to a file.

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

Data Structures

- struct [lslogging_queue_struct](#)
Our log object: time and message.

Macros

- #define [LSLOGGING_FILE_NAME](#) `"/tmp/pgpmac.log"`
Full name of the log file.
- #define [LSLOGGING_MSG_LENGTH](#) 2048
Fixed maximum length messages to keep some form of sanity.
- #define [LSLOGGING_QUEUE_LENGTH](#) 8192
Modest length queue.

Typedefs

- typedef struct
[lslogging_queue_struct](#) [lslogging_queue_t](#)
Our log object: time and message.

Functions

- void [lslogging_init](#) ()
Initialize the lslogging objects.
- void [lslogging_log_message](#) (char *fmt,...)
The routine everyone will be talking about.
- void * [lslogging_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

5.4.1 Detailed Description

Logs messages to a file.

Date

2012

Author

Keith Brister

Copyright

All Rights Reserved

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

5.4.2 Macro Definition Documentation

5.4.2.1 #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](#).

5.4.2.2 #define LSLOGGING_MSG_LENGTH 2048

Fixed maximum length messages to keep some form of sanity.

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

5.4.2.3 #define LSLOGGING_QUEUE_LENGTH 8192

Modest length queue.

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

5.4.3 Typedef Documentation

5.4.3.1 typedef struct lslogging_queue_struct lslogging_queue_t

Our log object: time and message.

5.4.4 Function Documentation

5.4.4.1 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");
    }

```

5.4.4.2 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);
    }

```

5.4.4.3 void lslogging_run ()

Start up the worker thread.

Definition at line 105 of file Islogging.c.

```

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

```

5.4.4.4 void* Islogging_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 Islogging.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);
        }
    }

```

5.4.5 Variable Documentation

5.4.5.1 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 Islogging.c.

5.4.5.2 FILE* lslogging_file [static]

our log file object

Definition at line 17 of file Islogging.c.

5.4.5.3 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 Islogging.c.

5.4.5.4 unsigned int lslogging_off = 0 [static]

next location to remove from the queue

Definition at line 34 of file lslogging.c.

5.4.5.5 unsigned int lslogging_on = 0 [static]

next location to add to the queue

Definition at line 33 of file lslogging.c.

5.4.5.6 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.

5.4.5.7 pthread_t lslogging_thread [static]

our thread

Definition at line 10 of file lslogging.c.

5.5 lspg.c File Reference

Postgresql support for the LS-CAT pgpmac project.

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

Data Structures

- struct [lspgQueryQueueStruct](#)
Store each query along with it's callback function.
- 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 `lspgQueryQueueStruct` `lspg_query_queue_t`
Store each query along with it's callback function.
- 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_init_motors_cb (lspg_query_queue_t *qqp, PGresult *pgr)`
Motor initialization callback.
- `void lspg_zoom_lut_cb (lspg_query_queue_t *qqp, PGresult *pgr)`
Zoom motor look up table callback.
- `void lspg_scint_lut_cb (lspg_query_queue_t *qqp, PGresult *pgr)`
- `void lspg_flight_lut_cb (lspg_query_queue_t *qqp, PGresult *pgr)`
Front Light Lookup table query callback Install the lookup table for the Front Light.
- `void lspg_blight_lut_cb (lspg_query_queue_t *qqp, PGresult *pgr)`
Back Light Lookup Table Callback Install the lookup table for the Back Light.
- `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 diffractionmeter lock.
- void `lspg_lock_diffractionmeter_wait ()`
Wait for the diffractionmeter lock.
- void `lspg_lock_diffractionmeter_done ()`
Finish up the lock diffractionmeter call.
- void `lspg_lock_diffractionmeter_all ()`
Convenience function that combines lock diffractionmeter 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_kvs_cb](#) ([lspg_query_queue_t](#) *qqp, PGresult *pgr)
retrieve kv pairs with new values
- 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 [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_nextshot_t](#) [lspg_nextshot](#)
the nextshot object
- [lspg_getcenter_t](#) [lspg_getcenter](#)
the getcenter object
- 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](#)

5.5.1 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](#).

5.5.2 Macro Definition Documentation

5.5.2.1 `#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 60 of file `lspg.c`.

5.5.2.2 `#define LS_PG_STATE_IDLE 1`

Definition at line 34 of file `lspg.c`.

5.5.2.3 `#define LS_PG_STATE_INIT -4`

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

5.5.2.4 `#define LS_PG_STATE_INIT_POLL -3`

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

5.5.2.5 `#define LS_PG_STATE_RECV 4`

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

5.5.2.6 `#define LS_PG_STATE_RESET -2`

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

5.5.2.7 `#define LS_PG_STATE_RESET_POLL -1`

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

5.5.2.8 `#define LS_PG_STATE_SEND 2`

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

5.5.2.9 `#define LS_PG_STATE_SEND_FLUSH 3`

Definition at line 36 of file `lspg.c`.

5.5.3 Typedef Documentation

5.5.3.1 `typedef struct lspg_lock_detector_struct lspg_lock_detector_t`

lock detector object Implements detector lock for exposure control

5.5.3.2 `typedef struct lspg_lock_diffractionmeter_struct lspg_lock_diffractionmeter_t`

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

5.5.3.3 typedef struct lspgQueryQueueStruct lspg_query_queue_t

Store each query along with it's callback function.

All calls are asynchronous

5.5.3.4 typedef struct lspg_seq_run_prep_struct lspg_seq_run_prep_t

Data collection running object.

5.5.3.5 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.

5.5.4 Function Documentation

5.5.4.1 char** lspg_array2ptrs (char * a)

returns a null terminated list of strings parsed from postgresql array

Definition at line 165 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] == 0 )
        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) {
        // TODO: print or otherwise log this condition
        // out of memory
        exit( 1);
    }

    //
    // allocate storage for the pointer array and the null terminator
    //
    rtn = (char **)calloc( n+1, sizeof( char *));
    if( rtn == NULL) {
        // TODO: print or otherwise log this condition
        // out of memory
        exit( 1);
    }
    rtni = 0;

    // Go through and create the individual strings
    sp = acums;
    *sp = 0;
    if( a[0] != '{') {
        // oh no! This isn't an array after all!
        // Zounds!
    }

```

```

    return NULL;
}
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;
                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.
//
rtn[rtni++] = strdup( acums);
rtn[rtni] = NULL;
return( rtn);
}

```

5.5.4.2 void lspg_blight_lut_cb (lspg_query_queue_t * qqp, PGresult * pgr)

Back Light Lookup Table Callback Install the lookup table for the Back Light.

Parameters

in	<i>qqp</i>	Our query
in	<i>pgr</i>	The query's result

Definition at line 428 of file lspg.c.

{

```

int i;

pthread_mutex_lock( &(blight->mutex));

blight->nlut = PQntuples( pgr)/2;
blight->lut = calloc( 2*blight->nlut, sizeof(double));
if( blight->lut == NULL) {
    lslogging_log_message( "Out of memory
        (lspg_blight_lut_cb)");
    pthread_mutex_unlock( &(blight->mutex));
    return;
}

for( i=0; i<PQntuples( pgr); i++) {
    blight->lut[i] = strtod( PQgetvalue( pgr, i, 0), NULL);
}

pthread_mutex_unlock( &(blight->mutex));
}

```

5.5.4.3 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 1144 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<PQntuples( pgr); i++) {
    sp = PQgetvalue( pgr, i, 0);
    if( sp != NULL && *sp != 0) {
        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()");
    }
}
}

```

5.5.4.4 void lspg.flight_lut.cb (lspg_query_queue_t * qqp, PGresult * pgr)

Front Light Lookup table query callback Install the lookup table for the Front Light.

Parameters

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

Definition at line 400 of file lspg.c.

```

{
int i;

```



```

pthread_mutex_lock( &(amp;flight->mutex));

flight->nlut = PQntuples( pgr)/2;
flight->lut = calloc( 2*flight->nlut, sizeof(double));
if( flight->lut == NULL) {
    lslogging_log_message( "Out of memory
(lspg_flight_lut_cb)");
    pthread_mutex_unlock( &(amp;flight->mutex));
    return;
}

for( i=0; i<PQntuples( pgr); i++) {
    flight->lut[i] = strtod( PQgetvalue( pgr, i, 0), NULL);
}

pthread_mutex_unlock( &(amp;flight->mutex));
}

```

5.5.4.5 void lspg_flush ()

Flush psql output buffer (ie, send the query)

Definition at line 1173 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;
    }
}

```

5.5.4.6 void lspg_getcenter_all ()

Convenience function to complete synchronous getcenter query.

Definition at line 1092 of file lspg.c.

```

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

```

5.5.4.7 void lspg_getcenter_call ()

Request a getcenter query.

Definition at line 1068 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() ");
}

```

5.5.4.8 void lspg_getcenter.cb (lspg_query_queue_t * qqp, PGresult * pgr)

Retrieve the data to center the crystal.

Definition at line 1003 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));
}

```

5.5.4.9 void lspg_getcenter_done ()

Done with getcenter query.

Definition at line 1086 of file lspg.c.

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

5.5.4.10 void lspg_getcenter_init ()

Initialize getcenter object.

Definition at line 1060 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);
}
```

5.5.4.11 void lspg_getcenter_wait ()

Wait for a getcenter query to return.

Definition at line 1078 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));
}
```

5.5.4.12 void lspg_init ()

Initiallize the lspg module.

Definition at line 1666 of file lspg.c.

```
{
    pthread_mutex_init( &lspg_queue_mutex, NULL);
    pthread_cond_init( &lspg_queue_cond, NULL);
    lspg_nextshot_init();
    lspg_getcenter_init();
    lspg_wait_for_detector_init();
    lspg_lock_diffractionmeter_init();
    lspg_lock_detector_init();
}
```

5.5.4.13 void lspg_init_motors_cb (lspg_query_queue_t * *qqp*, PGresult * *pgr*)

Motor initialization callback.

Parameters

in	<i>qqp</i>	The query queue item used to call us
in	<i>pgr</i>	The postgresql result object

Definition at line 284 of file lspg.c.

```

    {
    int i, j;
    uint32_t motor_number, motor_number_column, max_speed_column,
        max_accel_column, home_column;
    uint32_t units_column, coord_column, name_column, axis_column;
    uint32_t u2c_column;
    uint32_t format_column;
    uint32_t update_resolution_column;
    uint32_t update_format_column;
    char *sp;
    lspmac_motor_t *lsdp;

    name_column = PQfnumber( pgr, "mm_name");
    if( name_column == -1)
        return;

    motor_number_column = PQfnumber( pgr, "mm_motor");
    coord_column = PQfnumber( pgr, "mm_coord");
    units_column = PQfnumber( pgr, "mm_unit");
    axis_column = PQfnumber( pgr, "mm_axis");
    u2c_column = PQfnumber( pgr, "mm_u2c");
    format_column = PQfnumber( pgr, "mm_printf");
    max_speed_column = PQfnumber( pgr, "mm_max_speed");
    max_accel_column = PQfnumber( pgr, "mm_max_accel");
    update_resolution_column = PQfnumber( pgr, "mm_update_resolution");
    update_format_column = PQfnumber( pgr, "mm_update_format");
    home_column = PQfnumber( pgr, "mm_home");

    for( i=0; i<PQntuples( pgr); i++) {

        lsdp = NULL;
        for( j=0; j<lspmac_nmotors; j++) {
            if( strcmp(lspmac_motors[j].name, PQgetvalue( pgr, i,
                name_column)) == 0) {
                lsdp = &(lspmac_motors[j]);
                lsdp->motor_num = atoi(PQgetvalue( pgr, i,
                    motor_number_column));
                lsdp->coord_num = atoi( PQgetvalue( pgr, i,
                    coord_column));
                lsdp->units = strdup( PQgetvalue( pgr, i, units_column
                    ));
                lsdp->format = strdup( PQgetvalue( pgr, i,
                    format_column));
                // lsdp->u2c = atof(PQgetvalue( pgr, i, u2c_column));
                lsdp->max_speed = atof(PQgetvalue( pgr, i,
                    max_speed_column));
                lsdp->max_accel = atof(PQgetvalue( pgr, i,
                    max_accel_column));
                lsdp->update_resolution = atof(PQgetvalue( pgr, i,
                    update_resolution_column));
                lsdp->update_format = strdup( PQgetvalue( pgr, i,
                    update_format_column));

                if( PQgetisnull( pgr, i, axis_column))
                    lsdp->axis = NULL;
                else
                    lsdp->axis = strdup(PQgetvalue( pgr, i, axis_column));

                lsdp->home = lspg_array2ptrs(
                    PQgetvalue( pgr, i, home_column));

                lsdp->lspg_initialized = 1;
                break;
            }
        }
        if( lsdp == NULL)
            continue;

        /*
        if( fabs(lsdp->u2c) <= 1.0e-9)
            lsdp->u2c = 1.0;
        */
    }
}

```

5.5.4.14 void lspg_kv.cb (lspg_query_queue_t * qqp, PGresult * pgr)

retrieve kv pairs with new values

Parameters

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

Definition at line 1127 of file lspg.c.

```

    {
        int i;

        lslogging_log_message( "lspg_kvs_cb: %d tuples",
                               PQntuples(pgr));

        // Even i is key (the name)
        // Odd i is value
        //
        for( i=0; i<PQntuples(pgr)/2; i++) {
            lskvs_set( PQgetvalue( pgr, 2*i, 0), PQgetvalue( pgr, 2*i+1, 0));
        }
    }

```

5.5.4.15 void lspg_lock_detector_all ()

Detector lock convinence function.

Definition at line 915 of file lspg.c.

```

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

```

5.5.4.16 void lspg_lock_detector_call ()

Request (demand) a detector lock.

Definition at line 891 of file lspg.c.

```

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

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

```

5.5.4.17 void lspg_lock_detector_cb (lspg_query_queue_t * qqp, PGresult * pgr)

Callback for when the detector lock has be grabbed.

Definition at line 882 of file lspg.c.

```

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

```

5.5.4.18 void lspg_lock_detector_done ()

Finish waiting.

Definition at line 909 of file lspg.c.

```

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

```

5.5.4.19 void lspg_lock_detector_init ()

Initialize detector lock object.

Definition at line 874 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);
    }

```

5.5.4.20 void lspg_lock_detector_wait ()

Wait for the detector lock.

Definition at line 901 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));
    }

```

5.5.4.21 void lspg_lock_diffractionmeter_all ()

Convenience function that combines lock diffractionmeter calls.

Definition at line 856 of file lspg.c.

```

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

```

5.5.4.22 void lspg_lock_diffractionmeter_call ()

Request that the database grab the diffractionmeter lock.

Definition at line 832 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()");
    }

```

5.5.4.23 void lspg_lock_diffractionmeter_cb (lspg_query_queue_t * qqp, PGresult * pgr)

Callback routine for a lock diffractionmeter query.

Definition at line 823 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));
}

```

5.5.4.24 void lspg_lock_diffractionmeter_done ()

Finish up the lock diffractionmeter call.

Definition at line 850 of file lspg.c.

```

pthread_mutex_unlock( &(lspg_lock_diffractionmeter.
mutex));
}

```

5.5.4.25 void lspg_lock_diffractionmeter_init ()

initialize the diffractionmeter locking object

Definition at line 815 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);
}

```

5.5.4.26 void lspg_lock_diffractionmeter_wait ()

Wait for the diffractionmeter lock.

Definition at line 842 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));
}

```

5.5.4.27 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 1535 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;
}
}

```

5.5.4.28 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 1101 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);
}
}
}

```

5.5.4.29 void lspg_nextshot_call ()

Queue up a nextshot query.

Definition at line 715 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.nextshot()");
}

```

5.5.4.30 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 variable lspg_nextshot set.

Parameters

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

Definition at line 460 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.dsdir_isnull = PQgetisnull( pgr, 0,
    dsdir_c);
if( lspg_nextshot.dsdir != NULL)
    free( lspg_nextshot.dsdir);
lspg_nextshot.dsdir = strdup( PQgetvalue( pgr, 0, dsdir_c))
;

lspg_nextshot.dspid_isnull = PQgetisnull( pgr, 0,
    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));
}

```

5.5.4.31 void lspg_nextshot_done ()

Called when the next shot query has been processed.

Definition at line 733 of file lspg.c.

```

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

```

5.5.4.32 void lspg_nextshot_init ()

Initialize the nextshot variable, mutex, and condition.

Definition at line 707 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);
}

```

5.5.4.33 void lspg_nextshot_wait ()

Wait for the next shot query to get processed.

Definition at line 725 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));
}

```

5.5.4.34 PQnoticeProcessor lspg_notice_processor (void * arg, const char * msg)

Definition at line 1430 of file lspg.c.

```

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

```

5.5.4.35 void lspg_pg_connect ()

Connect to the pg server.

Definition at line 1436 of file lspg.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) {
        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( lspg_init_motors_cb, "
            select * from pmac.md2_getmotors()");
        lspg_query_push( NULL, "select pmac.md2_init()");
        lspg_query_push( lspg_zoom_lut_cb, "SELECT
            * FROM pmac.md2_zoom_lut()");
        lspg_query_push( lspg_flight_lut_cb, "
            SELECT * FROM pmac.md2_flight_lut()");
        lspg_query_push( lspg_blight_lut_cb, "
            SELECT * FROM pmac.md2_blight_lut()");
        lspg_query_push( lspg_scint_lut_cb,
            "SELECT * FROM pmac.md2_scint_lut()");

        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( lspg_init_motors_cb, "
            select * from pmac.md2_getmotors()");
        lspg_query_push( NULL, "select pmac.md2_init()");
        ls_pg_state = LS_PG_STATE_IDLE;
    }
    break;
}
}

```

5.5.4.36 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 1333 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;

        if( strstr( pgn->relnam, "_pmac") != NULL) {
            lspg_query_push( lspg_cmd_cb, "SELECT
pmac.md2_queue_next()");
        } else if( strstr( pgn->relnam, "_diff") != NULL) {
            lspg_query_push( lspg_nextaction_cb,
"SELECT action FROM px.nextaction()");
        } else if( strstr( pgn->relnam, "_kvs") != NULL) {
            lspg_query_push( lspg_kvs_cb, "SELECT
pmac.getkvs()");
        }
        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();
    }
}
}

```

5.5.4.37 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 79 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;
}

```

5.5.4.38 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 132 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 == lspg_query_queue_off
    ) {
        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);
};

```

5.5.4.39 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 103 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);
}

```

5.5.4.40 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 116 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;
}

```

5.5.4.41 void lspg_receive ()

Receive a result of a query.

Definition at line 1250 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->q, 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);
}
}
}

```

5.5.4.42 void lspg_run ()

Start 'er runnin'.

Definition at line 1678 of file lspg.c.

```

    {
        pthread_create( &lspg_thread, NULL, lspg_worker, NULL);
    }

```

5.5.4.43 void lspg_scint_lut.cb(lspg_query_queue_t * qqp, PGresult * pgr)

Parameters

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

Definition at line 376 of file lspg.c.

```

    {
        int i;
        pthread_mutex_lock( &(fscint->mutex));

        fscint->nlut = PQntuples( pgr)/2;
        fscint->lut = calloc( 2*fscint->nlut, sizeof( double));
        if( fscint->lut == NULL) {
            lslogging_log_message( "lspg_scint_lut.cb: Out of
                memory");
            pthread_mutex_unlock( &(fscint->mutex));
        }

        for( i=0; i<PQntuples( pgr); i++) {
            fscint->lut[i] = strtod( PQgetvalue( pgr, i, 0), NULL);
        }

        pthread_mutex_unlock( &(fscint->mutex));
    }

```

5.5.4.44 void lspg_send_next_query ()

send the next queued query to the DB server

Definition at line 1203 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;
    }
}
}

```

5.5.4.45 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 986 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();
}

```

5.5.4.46 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	ax	current alignment table x
in	ay	current alignment table y
in	az	current alignment table z

Definition at line 952 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);
}

```

5.5.4.47 void lspg_seq_run_prep.cb (lspg_query_queue_t * qqp, PGresult * pgr)

Callback for the seq_run_prep query.

Parameters

in	qqp	The query item that generated this callback
in	pgr	The result of the query

Definition at line 940 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));
}

```

5.5.4.48 void lspg_seq_run_prep.done ()

Indicate we are done waiting.

Definition at line 980 of file lspg.c.

```

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

```

5.5.4.49 void lspg_seq_run_prep.init ()

Initialize the data collection object.

Definition at line 932 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);
}

```

5.5.4.50 void lspg_seq_run_prep.wait ()

Wait for seq run prep query to return.

Definition at line 972 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));
}

```

5.5.4.51 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 1311 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));
}

```

5.5.4.52 void lspg_wait_for_detector.all ()

Combined call to wait for the detector.

Definition at line 796 of file lspg.c.

```

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

```

5.5.4.53 void lspg_wait_for_detector.call ()

initiate the wait for detector query

Definition at line 770 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()");
}

```

5.5.4.54 void lspg_wait_for_detector.cb (lspg_query_queue_t * qqp, PGresult * pgr)

Callback for the wait for detector query.

Definition at line 761 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
));
}

```

5.5.4.55 void lspg_wait_for_detector.done ()

Done waiting for the detector.

Definition at line 789 of file lspg.c.

```

pthread_mutex_unlock( &(lspg_wait_for_detector.mutex
));
}

```

5.5.4.56 void lspg_wait_for_detector.init ()

initialize the detector timing object

Definition at line 753 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);
}

```

5.5.4.57 void lspg_wait_for_detector.wait ()

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

Definition at line 781 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));
}

```

5.5.4.58 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 1586 of file lspg.c.

```

    {
static struct pollfd fda[2]; // 0=signal handler, 1=pg socket
static int nfda = 0;
static sigset_t our_sigset;
int sigfd;

sigemptyset( &our_sigset);
sigaddset( &our_sigset, SIGUSR1);

//
// block ordinary signal mechanism
//
sigprocmask(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 conneced
//
lspgfd.fd = -1;

while( 1) {
    int pollrtn;
    int poll_timeout_ms;

    lspg_next_state();

    if( lspgfd.fd == -1) {
        //
        // Here a connection to the database is not established.
        // Periodically try again. Should possibly arrange to reconnect
        // to signalfd but that's unlikely to be 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--;
    }
}
}

```


5.5.4.59 void `lspg_zoom_lut_cb` (`lspg_query_queue_t` * *qqp*, `PGresult` * *pgr*)

Zoom motor look up table callback.

Parameters

in	<i>qqp</i>	the queue item responsible for calling us
in	<i>pgr</i>	The Postgresql result object

Definition at line 353 of file `lspg.c`.

```

    {
        int i;

        pthread_mutex_lock( &(zoom->mutex));

        zoom->nlut = PQntuples( pgr)/2;
        zoom->lut = calloc( 2*zoom->nlut, sizeof(double));
        if( zoom->lut == NULL) {
            lslogging_log_message( "Out of memory
                                   (lspg_zoom_lut_cb)");
            pthread_mutex_unlock( &(zoom->mutex));
            return;
        }

        for( i=0; i<PQntuples( pgr); i++) {
            zoom->lut[i] = strtod( PQgetvalue( pgr, i, 0), NULL);
        }

        pthread_mutex_unlock( &(zoom->mutex));
    }

```

5.5.5 Variable Documentation

5.5.5.1 int `ls_pg_state` = `LS_PG_STATE_INIT` [static]

State of the lspg state machine.

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

5.5.5.2 `PostgresPollingStatusType` `lspg_connectPoll_response` [static]

Used to determine state while connecting.

Definition at line 69 of file `lspg.c`.

5.5.5.3 `lspg_getcenter_t` `lspg_getcenter`

the getcenter object

Definition at line 73 of file `lspg.c`.

5.5.5.4 `lspg_lock_detector_t` `lspg_lock_detector` [static]

Definition at line 870 of file `lspg.c`.

5.5.5.5 `lspg_lock_diffractionmeter_t` `lspg_lock_diffractionmeter` [static]

Definition at line 811 of file `lspg.c`.

5.5.5.6 `lspg_nextshot_t lspg_nextshot`

the nextshot object

Definition at line 72 of file `lspg.c`.

5.5.5.7 `lspg_query_queue_t lspg_query_queue[LS_PG_QUERY_QUEUE_LENGTH]` `[static]`

Our query queue.

Definition at line 61 of file `lspg.c`.

5.5.5.8 `unsigned int lspg_query_queue_off = 0` `[static]`

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

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

5.5.5.9 `unsigned int lspg_query_queue_on = 0` `[static]`

Next position to add something to the queue.

Definition at line 62 of file `lspg.c`.

5.5.5.10 `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 64 of file `lspg.c`.

5.5.5.11 `pthread_cond_t lspg_queue_cond` `[static]`

keeps the queue from overflowing

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

5.5.5.12 `pthread_mutex_t lspg_queue_mutex` `[static]`

keep the queue from getting tangled

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

5.5.5.13 `PostgresPollingStatusType lspg_resetPoll_response` `[static]`

Used to determine state while reconnecting.

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

5.5.5.14 `lspg_seq_run_prep_t lspg_seq_run_prep` `[static]`

Definition at line 928 of file `lspg.c`.

5.5.5.15 pthread_t lspg_thread [static]

our worker thread

Definition at line 42 of file lspg.c.

5.5.5.16 lspg_wait_for_detector_t lspg_wait_for_detector [static]

Instance of the detector timing object.

Definition at line 749 of file lspg.c.

5.5.5.17 struct pollfd lspgfd [static]

our poll info

Definition at line 45 of file lspg.c.

5.5.5.18 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.

5.5.5.19 PGconn* q = NULL [static]

Database connector.

Definition at line 68 of file lspg.c.

5.6 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.

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 20000.0`
Minimum time between commands to the pmac.
- `#define PMAC_CMD_QUEUE_LENGTH 2048`
Size of the PMAC command queue.

Typedefs

- `typedef struct md2StatusStruct md2_status_t`
The block of memory retrieved in a status request.

Functions

- `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.

- `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, unsigned char *data, void(*responseCB)(pmac_cmd_queue_t *, int, unsigned char *), int no_reply)`
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 (unsigned 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, unsigned char *buff)`
Receive a reply that does not require multiple buffers.
- `void lspmac_SendControlReplyPrintCB (pmac_cmd_queue_t *cmd, int nreceived, unsigned 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, unsigned 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 *fmt,...)`
Send a one line command.
- `pmac_cmd_queue_t * lspmac_SockSendline_nr (char *fmt,...)`
Send a command and ignore the response.
- `pmac_cmd_queue_t * lspmac_SockSendControlCharPrint (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.
- `void lspmac_get_status_cb (pmac_cmd_queue_t *cmd, int nreceived, unsigned 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_GetAllIVarsCB` (`pmac_cmd_queue_t` *cmd, int nreceived, unsigned 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` (`pmac_cmd_queue_t` *cmd, int nreceived, unsigned 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` (void(*responseCB)(`pmac_cmd_queue_t` *, int, unsigned 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 *name)
Move a given motor to one of its preset positions.
- 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.
- `lspmac_motor_t` * `lspmac_motor_init` (`lspmac_motor_t` *d, int motor_number, 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)
Inititalize 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, double scale, 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, double scale, 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_newKV_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.
- `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 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](#) [16]
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 * blight_ud`
- Back light Up/Down actuator.
- `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_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_bi_t * cryo_switch`
- that little toggle switch for the cryo
- static int `linesReceived` = 0
- current number of lines received
- 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.

5.6.1 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](#).

5.6.2 Macro Definition Documentation

5.6.2.1 `#define LS_PMAC_STATE_CR 7`

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

5.6.2.2 `#define LS_PMAC_STATE_DETACHED 0`

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

5.6.2.3 `#define LS_PMAC_STATE_GB 10`

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

5.6.2.4 `#define LS_PMAC_STATE_GMR 6`

Definition at line 44 of file lspmac.c.

5.6.2.5 `#define LS_PMAC_STATE_IDLE 1`

Definition at line 39 of file lspmac.c.

5.6.2.6 `#define LS_PMAC_STATE_RESET -1`

Definition at line 37 of file lspmac.c.

5.6.2.7 `#define LS_PMAC_STATE_RR 8`

Definition at line 46 of file lspmac.c.

5.6.2.8 `#define LS_PMAC_STATE_SC 2`

Definition at line 40 of file lspmac.c.

5.6.2.9 `#define LS_PMAC_STATE_WACK 5`

Definition at line 43 of file lspmac.c.

5.6.2.10 `#define LS_PMAC_STATE_WACK_CC 4`

Definition at line 42 of file lspmac.c.

5.6.2.11 `#define LS_PMAC_STATE_WACK_NFR 3`

Definition at line 41 of file lspmac.c.

5.6.2.12 `#define LS_PMAC_STATE_WACK_RR 9`

Definition at line 47 of file lspmac.c.

5.6.2.13 `#define LS_PMAC_STATE_WCR 11`

Definition at line 49 of file lspmac.c.

5.6.2.14 `#define LS_PMAC_STATE_WGB 12`

Definition at line 50 of file lspmac.c.

5.6.2.15 `#define LSPMAC_PRESET_REGEX "(.*\\.%s\\.presets)\\.([0-9]+)\\.((name|position))"`

Regex to pick out preset name and corresponding position.

Definition at line 112 of file lspmac.c.

5.6.2.16 #define PMAC_CMD_QUEUE_LENGTH 2048

Size of the PMAC command queue.
Definition at line 156 of file lspmac.c.

5.6.2.17 #define pmac_cmd_size 8

PMAC command size in bytes.
Definition at line 122 of file lspmac.c.

5.6.2.18 #define PMAC_MIN_CMD_TIME 20000.0

Minimum time between commands to the pmac.
Definition at line 152 of file lspmac.c.

5.6.2.19 #define PMACPORT 1025

The PMAC (only) listens on this port.
Definition at line 116 of file lspmac.c.

5.6.2.20 #define VR_CTRL_RESPONSE 0xc4

Definition at line 138 of file lspmac.c.

5.6.2.21 #define VR_DOWNLOAD 0x40

Definition at line 125 of file lspmac.c.

5.6.2.22 #define VR_FWDOWNLOAD 0xcb

Definition at line 142 of file lspmac.c.

5.6.2.23 #define VR_IPADDRESS 0xe0

Definition at line 143 of file lspmac.c.

5.6.2.24 #define VR_PMAC_FLUSH 0xb3

Definition at line 129 of file lspmac.c.

5.6.2.25 #define VR_PMAC_GETBUFFER 0xc5

Definition at line 139 of file lspmac.c.

5.6.2.26 #define VR_PMAC_GETLINE 0xb1

Definition at line 128 of file lspmac.c.

5.6.2.27 #define VR_PMAC_GETMEM 0xb4

Definition at line 130 of file lspmac.c.

5.6.2.28 #define VR_PMAC_GETRESPONSE 0xbf

Definition at line 136 of file lspmac.c.

5.6.2.29 #define VR_PMAC_PORT 0xbe

Definition at line 135 of file lspmac.c.

5.6.2.30 #define VR_PMAC_READREADY 0xc2

Definition at line 137 of file lspmac.c.

5.6.2.31 #define VR_PMAC_SENDCTRLCHAR 0xb6

Definition at line 132 of file lspmac.c.

5.6.2.32 #define VR_PMAC_SENDLINE 0xb0

Definition at line 127 of file lspmac.c.

5.6.2.33 #define VR_PMAC_SETBIT 0xba

Definition at line 133 of file lspmac.c.

5.6.2.34 #define VR_PMAC_SETBITS 0xbb

Definition at line 134 of file lspmac.c.

5.6.2.35 #define VR_PMAC_SETMEM 0xb5

Definition at line 131 of file lspmac.c.

5.6.2.36 #define VR_PMAC_WRITEBUFFER 0xc6

Definition at line 140 of file lspmac.c.

5.6.2.37 #define VR_PMAC_WRITEERROR 0xc7

Definition at line 141 of file lspmac.c.

5.6.2.38 #define VR_UPLOAD 0xc0

Definition at line 124 of file lspmac.c.

5.6.3 Typedef Documentation

5.6.3.1 typedef struct md2StatusStruct md2_status_t

The block of memory retrieved in a status request.

5.6.4 Function Documentation

5.6.4.1 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 449 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);
}

```

5.6.4.2 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 421 of file lspmac.c.

```

{
    int i;          // row counter
    int j;          // column counter

    pthread_mutex_lock( &ncurses_mutex);

    for( i=0; n > 0; i++) {
        for( j=0; j<16 && n > 0; j++) {
            if( j==8)
                wprintw( term_output, " ");
            wprintw( term_output, " %02x", *(s + 16*i + j));
            n--;
        }
        wprintw( term_output, "\n");
    }
    wprintw( term_output, "\n");

    pthread_mutex_unlock( &ncurses_mutex);
}

```

5.6.4.3 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 470 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;
    }

```

5.6.4.4 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 2716 of file lspmac.c.

```

    {
        blight->moveAbs( blight, 0.0);
    }

```

5.6.4.5 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 2707 of file lspmac.c.

```

    {
        int z;

        blight->moveAbs( blight, lspmac_getPosition
            ( zoom));
    }

```

5.6.4.6 lspmac_bi_t* lspmac_bi.init(lspmac_bi_t * d, int * ptr, int mask, char * onEvent, char * offEvent)

Initialize binary input.

Definition at line 2564 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;
    }

```

5.6.4.7 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 2461 of file lspmac.c.

```

    {
        lspmac_nmotors++;

        lskvs_regcomp( &(d->preset_regex), REG_EXTENDED,
            LSPMAC_PRESET_REGEX, name);
        d->presets = NULL;
        d->name = strdup( name);
        d->moveAbs = lspmac_moveabs_bo_queue;
        d->read = lspmac_bo_read;
        d->lut = NULL;
    }

```



```

d->n lut          = 0;
d->actual_pos_cnts_p = NULL;
d->status1_p      = NULL;
d->status2_p      = NULL;
d->motor_num      = -1;
d->dac_mvar       = NULL;
d->win            = NULL;
d->write_fmt      = strdup( write_fmt);
d->read_ptr       = read_ptr;
d->read_mask      = read_mask;
d->homing         = 0;
d->win            = NULL;
d->u2c            = lsredis_get_obj( "%s.u2c", name);

d->lspg_initialized = 0;
return d;
}

```

5.6.4.8 void Ispmac_bo_read (Ispmac_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	mp	The motor
----	----	-----------

Definition at line 1012 of file Ispmac.c.

```

{
char s[512];
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;

// Not sure what kind of status makes sense to report
mp->statuss[0] = 0;
pthread_mutex_unlock( &(mp->mutex));

if( changed)
    lsevents_send_event( "%s %d", mp->name, pos);
}

```

5.6.4.9 void Ispmac_cryoSwitchChanged_cb (char * event)

Definition at line 2669 of file Ispmac.c.

```

{
int pos;

pthread_mutex_lock( &(cryo->mutex));
pos = cryo->position;
pthread_mutex_unlock( &(cryo->mutex));

cryo->moveAbs( cryo, pos ? 0.0 : 1.0);
}

```

5.6.4.10 Ispmac_motor_t* Ispmac_dac_init (Ispmac_motor_t * d, int * posp, double scale, char * mvar, char * name, void (*)(Ispmac_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>scale</i>	Scale factor (units)
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 2501 of file lspmac.c.

```

{
    lspmac_nmotors++;
    lskvs_regcomp( &(d->preset_regex), REG_EXTENDED,
        LSPMAC_PRESET_REGEX, name);
    d->presets = NULL;

    d->name = strdup( name);
    d->moveAbs = moveAbs;
    d->read = lspmac_dac_read;
    d->lut = NULL;
    d->nlut = 0;
    d->actual_pos_cnts_p = posp;
    d->status1_p = NULL;
    d->status2_p = NULL;
    d->motor_num = -1;
    d->dac_mvar = strdup(mvar);
    d->u2c = lsredis_get_obj( "%s.u2c", name);
    d->homing = 0;
    d->win = NULL;

    d->lspg_initialized = 0;
    return d;
}

```

5.6.4.11 void lspmac_dac_read (lspmac_motor_t * mp)

Read a DAC motor position.

Parameters

in	<i>mp</i>	The motor
----	-----------	-----------

Definition at line 1035 of file lspmac.c.

```

{
    int pos;
    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;
        }
    }

    // Not sure what kind of status makes sense to report
    mp->statusss[0] = 0;

    pthread_mutex_unlock( &(mp->mutex));
}

```

5.6.4.12 void lspmac_Error (unsigned 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 667 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();
}

```

5.6.4.13 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 2429 of file lspmac.c.

```

{
    lspmac_nmotors++;

    d->presets = NULL;
    d->name = strdup("fastShutter");
    d->u2c = lsredis_get_obj( "%s.u2c", d->name );
};
lskvs_regcomp( &(d->preset_regex), REG_EXTENDED,
    LSPMAC_PRESET_REGEX, d->name);
d->moveAbs = lspmac_moveabs_fshut_queue
    ;
d->read = lspmac_shutter_read;
d->lut = NULL;
d->nlut = 0;
d->actual_pos_cnts_p = NULL;
d->status1_p = NULL;
d->status2_p = NULL;
d->motor_num = -1;
d->dac_mvar = NULL;
d->homing = 0;
d->win = NULL;

d->lspg_initialized = 0;
return d;
}

```

5.6.4.14 void lspmac_get_status ()

Request a status update from the PMAC.

Definition at line 1606 of file lspmac.c.

```

{
    lspmac_send_command( VR_UPLOAD, VR_PMAC_GETMEM
        , 0x400, 0, sizeof(md2_status_t), NULL, lspmac_get_status_cb
        , 0);
}

```

5.6.4.15 void lspmac_get_status.cb (pmac_cmd_queue_t * cmd, int nreceived, unsigned 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 1434 of file lspmac.c.

```

{
    static int cnt = 0;
    static char s[256];
    static struct timeval ts1, ts2;

    char *sp;
    int i, pos;
    lspmac_motor_t *mp;
    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));
    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));

        pos = (*(bp->ptr) & bp->mask) == 0 ? 0 : 1;
    }
}

```

```

    if( bp->first_time) {
        bp->first_time = 0;
        if( pos==1 && bp->changeEventOn != NULL && bp->changeEventOn
            [0] != 0)
            lsevents_send_event( lspmac_bis[i].
            changeEventOn);
        if( pos==0 && bp->changeEventOff != NULL && bp->
            changeEventOff[0] != 0)
            lsevents_send_event( lspmac_bis[i].
            changeEventOff);
    } else {
        if( pos != bp->previous) {
            if( pos==1 && bp->changeEventOn != NULL && bp->
                changeEventOn[0] != 0)
                lsevents_send_event( lspmac_bis[i].
                changeEventOn);
            if( pos==0 && bp->changeEventOff != NULL && bp->
                changeEventOff[0] != 0)
                lsevents_send_event( lspmac_bis[i].
                changeEventOff);
        }
    }
    bp->previous = pos;
    pthread_mutex_unlock( &(bp->mutex));
}

pthread_mutex_lock( &ncurses_mutex);

// acc11c_1
// mask bit
// 0x01 0 Air pressure OK
// 0x02 1 Air bearing OK
// 0x04 2 Cryo switch
// 0x08 3
// 0x10 4
// 0x20 5
// 0x40 6 Cryo is back

//
// acc11c_2
// mask bit
// 0x01 0 Fluor Dector back
// 0x02 1 Sample Detected
// 0x04 2
// 0x08 3
// 0x10 4
// 0x20 5 Etel Ready
// 0x40 6 Etel On
// 0x80 7 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
// mask bit
// 0x01 0 Minikappa OK
// 0x02 1
// 0x04 2
// 0x08 3 Arm Parked

// acc11c_5
// mask bit
// 0x01 0 Mag Off
// 0x02 1 Condenser Out
// 0x04 2 Cryo Back
// 0x08 3 Dryer On
// 0x10 4 FluoDet Out
// 0x20 5

```

```

// 0x40 6 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
// mask bit
// 0x0080 7 Etel Enable
// 0x0100 8 Fast Shutter Enable
// 0x0200 9 Fast Shutter Manual Enable
// 0x0400 10 Fast Shutter On

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;
}
*/
}

```

5.6.4.16 void lspmac.GetAllIVars ()

Request the values of all the I variables.

Definition at line 1631 of file lspmac.c.

```

{
    static char *cmds = "IO..8191";
    lspmac_send_command( VR_DOWNLOAD,
        VR_PMAC_SENDLINE, 0, 0, strlen( cmds), cmds,
        lspmac_GetAllIVarsCB, 0);
}

```

5.6.4.17 void lspmac.GetAllIVarsCB (pmac_cmd_queue_t * cmd, int nreceived, unsigned 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 1614 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);
}
}

```

5.6.4.18 void Ispmac_GetAllMVars ()

Request the values of all the M variables.

Definition at line 1656 of file Ispmac.c.

```

    {
static char *cmds = "M0..8191->";
lspmac_send_command( VR_DOWNLOAD,
VR_PMAC_SENDLINE, 0, 0, strlen( cmds), cmds,
lspmac_GetAllMVarsCB, 0);
}

```

5.6.4.19 void Ispmac_GetAllMVarsCB (pmac_cmd_queue_t * cmd, int nreceived, unsigned 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 1639 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);
}
}

```

5.6.4.20 void Ispmac_Getmem ()

Request a block of double buffer memory.

Definition at line 1003 of file Ispmac.c.

```

    {
int nbytes;
nbytes = (dbmemIn + 1400 > sizeof( dbmem)) ? sizeof( dbmem)
- dbmemIn : 1400;
lspmac_SockGetmem( dbmemIn, nbytes);
}

```

5.6.4.21 void Ispmac_GetmemReplyCB (pmac_cmd_queue_t * cmd, int nreceived, unsigned char * buff)

Service a reply to the getmem command.

Not currently used.

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 934 of file lspmacc.c.

```

{
    memcpy( &(dbmem[ntohs(cmd->pcmd.wValue)]), buff, nreceived);
    dbmemIn += nreceived;
    if( dbmemIn >= sizeof( dbmem)) {
        dbmemIn = 0;
    }
}

```

5.6.4.22 double lspmacc_getPosition (lspmacc_motor_t * mp)

get the motor position (with locking)

Parameters

<i>mp</i>	the motor object
-----------	------------------

Definition at line 1230 of file lspmacc.c.

```

{
    double rtn;
    pthread_mutex_lock( &(mp->mutex));
    rtn = mp->position;
    pthread_mutex_unlock( &(mp->mutex));
    return rtn;
}

```

5.6.4.23 void lspmacc_GetShortReplyCB (pmac_cmd_queue_t * cmd, int nreceived, unsigned 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 876 of file lspmacc.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));
}

```

5.6.4.24 void lspmac_home1_queue(lspmac_motor_t * mp)

Home the motor.

Parameters

in	<i>mp</i>	motor we are concerned about
----	-----------	------------------------------

Definition at line 1106 of file lspmac.c.

```

{
char openloops[32];
char *sp;
int i;

pthread_mutex_lock( &(mp->mutex));

// We got here before the initialization routine finished
// TODO: arrange to retry or at least indicated we haven't run
//
if( (mp->lspg_initialized & 1) == 0) {
    pthread_mutex_unlock( &(mp->mutex));
    return;
}

// Each of the motors should have this defined
// but let's not seg fault if home is missing
//
if( mp->home == NULL || *(mp->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.
//
if( mp->coord_num > 0) {
    for( i=0; i<lspmac_nmotors; i++) {
        if( &(lspmac_motors[i]) == mp)
            continue;
        if( lspmac_motors[i].coord_num == mp->coord_num) {
            if( lspmac_motors[i].homing) {
                pthread_mutex_unlock( &(mp->mutex));
                return;
            }
        }
    }
}
mp->homing = 1;

// 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) {
    snprintf( openloops, sizeof(openloops)-1, "%d$", mp->motor_num);
    openloops[sizeof(openloops)-1] = 0;
    lspmac_SockSendline( openloops);
}

pthread_mutex_unlock( &(mp->mutex));
}

```

5.6.4.25 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 1187 of file lspmac.c.

```

{

char **spp;

//
// At this point we are in open loop.
// Run the motor specific commands
//
pthread_mutex_lock( &(mp->mutex));
//
// 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( mp->home == NULL || mp->homing != 1) {
    pthread_mutex_unlock( &(mp->mutex));
    return;
}

for( spp = mp->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_SockSendline( *spp);
}

mp->homing = 2;
pthread_mutex_unlock( &(mp->mutex));
}

```

5.6.4.26 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 2578 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);

    //
    // Initialize the motor objects
    //

    p = &md2_status;

    omega = lspmac_motor_init( &(lspmac_motors
    [ 0]), 1, 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]), 2, 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]), 3, 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]), 4, 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]), 5, 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]), 6, 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]), 7, 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]), 8, 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]), 9, 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]), 10, 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]), 11, 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]), 17, 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]), 18, 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]), 19, 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]), 20, 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, 160.0, "M1200", "frontLight.intensity",
    lspmac_movedac_queue);
    blight = lspmac_dac_init( &(lspmac_motors[1
    7]), &p->back_dac, 160.0, "M1201", "backLight.intensity",
    lspmac_movedac_queue);

```

```

fscint = lspmac_dac_init( &(lspmac_motors[1
8]), &p->scint_piezo, 320.0, "M1203", "scint.focus",
lspmac_movedac_queue);

blight_ud = lspmac_bo_init( &(lspmac_motors
[19]), "backLight", "M1101=%d", &(md2_status.accl1c_5), 0x02)
;
cryo = lspmac_bo_init( &(lspmac_motors[20
]), "cryo", "M1102=%d", &(md2_status.accl1c_5), 0x04);
dryer = lspmac_bo_init( &(lspmac_motors[2
1]), "dryer", "M1103=%d", &(md2_status.accl1c_5), 0x08);
fluo = lspmac_bo_init( &(lspmac_motors[22
]), "fluo", "M1008=%d", &(md2_status.accl1c_2), 0x01);
flight_oo = lspmac_soft_motor_init( &(
lspmac_motors[23]), "frontLight", 1.0,
lspmac_moveabs_frontlight_oo_queue);
blight_f = lspmac_soft_motor_init( &(
lspmac_motors[24]), "backLight.factor", 1.0,
lspmac_moveabs_blight_factor_queue);
flight_f = lspmac_soft_motor_init( &(
lspmac_motors[25]), "frontLight.factor", 1.0,
lspmac_moveabs_flight_factor_queue);

cryo_switch = lspmac_bi_init( &(lspmac_bis
[0]), &(md2_status.accl1c_1), 0x04, "CryoSwitchChanged", "
CryoSwitchChanged");

//
// 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);
}

```

5.6.4.27 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 2308 of file lspmac.c.

```

    {
        lspmac_move_or_jog_abs_queue( mp,
            requested_position, 1);
    }

```

5.6.4.28 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 2723 of file lspmac.c.

```

    {
        double z;

        z = lspmac_getPosition( zoom);
        if( lspmac_getPosition( flight_oo) != 0.0) {
            flight->moveAbs( flight, z);
        } else {
            flight->moveAbs( flight, 0.0);
        }
        if( lspmac_getPosition( blight_ud) != 0.0) {
            blight->moveAbs( blight, z);
        } else {
            blight->moveAbs( blight, 0.0);
        }
    }
}

```

5.6.4.29 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.

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 308 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;
}

```

5.6.4.30 `lspmac_motor_t* lspmac_motor_init(lspmac_motor_t *d, int motor_number, 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>motor_number</i>	The PMAC motor number
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>	Name of this motor (to match database)
in	<i>moveAbs</i>	Method to use to move this motor

Definition at line 2386 of file `lspmac.c`.

```

{
    lspmac_nmotors++;

    pthread_mutex_init( &(d->mutex), NULL);
    pthread_cond_init( &(d->cond), NULL);

    lskvs_regcomp( &(d->preset_regex), REG_EXTENDED,
        LSPMAC_PRESET_REGEX, name);

    d->u2c = lsredis_get_obj( "%s.u2c", name);
    d->presets = NULL;
    d->name = strdup(name);
    d->moveAbs = moveAbs;
    d->read = lspmac_pmacmotor_read;
    d->lut = NULL;
    d->nlut = 0;
    d->actual_pos_cnts_p = posp;
    d->status1_p = stat1p;
    d->status2_p = stat2p;
    d->motor_num = motor_number;
    d->dac_mvar = NULL;
}

```

```

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);
d->homing      = 0;
d->lspg_initialized = 0;

return d;
}

```

5.6.4.31 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.

< buffer to send to pmac

< 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 2162 of file lspmac.c.

```

{
char s[512];
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);

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->requested_pos_cnts = u2c * requested_position;
requested_pos_cnts = mp->requested_pos_cnts;
coord_num = mp->coord_num;
motor_num = mp->motor_num;

if( use_jog || mp->axis == NULL || *(mp->axis) == 0) {
    use_jog = 1;
} else {
    use_jog = 0;
    axis = *(mp->axis);
    q100 = 1 << (mp->coord_num -1);
}

pthread_mutex_unlock( &(mp->mutex));

if( use_jog) {
    snprintf( s, sizeof(s)-1, "##d j=%d", motor_num, requested_pos_cnts);
} else {

    //
    // 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_SockSendline( "M5075=(M5075 | %d)", q100);

switch( axis) {
case 'A':
    snprintf( s, sizeof(s)-1, "%d Q16=%d Q100=%d B146R", coord_num,
        requested_pos_cnts, q100);
    break;

case 'B':
    snprintf( s, sizeof(s)-1, "%d Q17=%d Q100=%d B147R", coord_num,
        requested_pos_cnts, q100);
    break;

case 'C':
    snprintf( s, sizeof(s)-1, "%d Q18=%d Q100=%d B148R", coord_num,
        requested_pos_cnts, q100);
    break;

case 'X':
    snprintf( s, sizeof(s)-1, "%d Q10=%d Q100=%d B140R", coord_num,
        requested_pos_cnts, q100);
    break;

case 'Y':
    snprintf( s, sizeof(s)-1, "%d Q11=%d Q100=%d B141R", coord_num,
        requested_pos_cnts, q100);
    break;

case 'Z':
    snprintf( s, sizeof(s)-1, "%d Q12=%d Q100=%d B142R", coord_num,
        requested_pos_cnts, q100);
    break;

case 'U':
    snprintf( s, sizeof(s)-1, "%d Q13=%d Q100=%d B143R", coord_num,
        requested_pos_cnts, q100);
    break;

case 'V':
    snprintf( s, sizeof(s)-1, "%d Q14=%d Q100=%d B144R", coord_num,
        requested_pos_cnts, q100);
    break;

case 'W':
    snprintf( s, sizeof(s)-1, "%d Q15=%d Q100=%d B145R", coord_num,
        requested_pos_cnts, q100);
    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));
mp->pq = lspmac_SockSendline_nr( s);
pthread_mutex_unlock( &(mp->mutex));
}

```


5.6.4.32 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 2277 of file lspmac.c.

```

{
double pos;
int err;

if( preset == NULL || *preset == 0)
    return;

pthread_mutex_lock( &(mp->mutex));
pos = lskvs_find_preset_position( mp, preset, &err)
;
pthread_mutex_unlock( &(mp->mutex));

lspmac_move_or_jog_abs_queue( mp, pos, use_jog);
}

```

5.6.4.33 void lspmac_move_preset_queue (lspmac_motor_t * mp, char * 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

< 0 = stns.2.appy.preset, for example, 1 = index, 2 = "position" or "name"

< 0 = stns.2.appy.preset, for example, 1 = index, 2 = "position" or "name"

Definition at line 1911 of file lspmac.c.

```

{

lskvs_kvs_list_t *q, *r;
regmatch_t q_pmatch[4];
regmatch_t r_pmatch[4];
double pos;

lslogging_log_message( "lspmac_move_preset_queue: Called
    with motor %s and preset named '%s'", mp->name, name);

//
// This checks both the ".name" and the ".position" entries
// but as long as no one gives names like "1.23" to their presets
// we should be OK.
//
for( q=mp->presets; q != NULL; q = q->next) {
    if( strcmp( name, q->kvs->v) == 0)
        break;
}
if( q == NULL) {
    lslogging_log_message( "lspmac_move_preset_queue: no
        preset named %s found for motor %s", name, mp->name);
    return;
}
if( regex( &(mp->preset_regex), q->kvs->k, 4, q_pmatch, 0)
    != 0 || q_pmatch[2].rm_so == -1 || q_pmatch[2].rm_eo == -1) {
    lslogging_log_message( "lspmac_move_preset_queue:
        Could not parse %s (q)", q->kvs->k);
    return;
}
}

```

```

}

//
// find the position entry. Note we are assuming that we've already found
// the name and only the position is left with the sample index
//
for( r=mp->presets; r != NULL; r = r->next) {
    if( r == q)
        continue;
    if( regexec( &(mp->preset_regex), r->kvs->k, 4, r_pmatch, 0
    ) != 0 || r_pmatch[2].rm_so == -1 || r_pmatch[2].rm_eo == -1) {
        lslogging_log_message( "lspmac_move_preset_queue:
        Could not parse %s (r)", r->kvs->k);
        return;
    }

    //
    // Make sure everything matches up to (and through) the array index
    //
    if( strcmp( q->kvs->k, r->kvs->k, q_pmatch[2].rm_eo + 1) == 0) {
        break;
    }
}

if( r == NULL) {
    lslogging_log_message( "lspmac_move_preset_queue:
    Could not find position for preset '%s' for motor '%s'", name, mp->name);
    return;
}

errno = 0;
pos = strtod( r->kvs->v, NULL);
if( errno != 0) {
    lslogging_log_message( "lspmac_move_preset_queue:
    Could not parse preset position '%s' for motor '%s'", r->kvs->v, mp->name);
    return;
}
mp->moveAbs( mp, pos);
lslogging_log_message( "lspmac_move_preset_queue: moving
    %s to preset '%s' (%f)", mp->name, name, pos);
}

```

5.6.4.34 void lspmac.moveabs.blight_factor.queue (lspmac_motor_t * mp, double pos)

Definition at line 2114 of file lspmac.c.

```

{

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));
    lsredis_setstr( blight->u2c, lsredis_getstr
    (blight->format), pos / 100.0);
    pthread_mutex_unlock( &(blight->mutex));

    blight->moveAbs( blight, lspmac_getPosition
    ( zoom));
}
}

```

5.6.4.35 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 2005 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;
mp->pq = lspmac_SockSendline_nr( mp->write_fmt
    , mp->requested_pos_cnts);

pthread_mutex_unlock( &(mp->mutex));
}

```

5.6.4.36 void lspmac.moveabs.flight_factor.queue (lspmac_motor_t * mp, double pos)

Definition at line 2096 of file lspmac.c.

```

{

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));

lsredis_setstr( flight->u2c, lsredis_getstr
    (flight->format), pos / 100.0);

pthread_mutex_unlock( &(flight->mutex));

flight->moveAbs( flight, lspmac_getPosition
    ( zoom));
}
}

```

5.6.4.37 void lspmac.moveabs.frontlight.oo.queue (lspmac_motor_t * mp, double pos)

"move" frontlight on/off

Definition at line 2084 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));
}
}

```

5.6.4.38 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 1978 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
    //
    mp->pq = lspmac_SockSendline_nr( "M1124=0 M1125=1
    M1126=1");
} else {
    //
    // ManualOn=0, ManualEnable=0, ScanEnable=1
    //
    mp->pq = lspmac_SockSendline_nr( "M1126=0 M1125=0
    M1124=1");
}

pthread_mutex_unlock( &(mp->mutex));
}

```

5.6.4.39 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 2298 of file lspmac.c.

```

    {

lspmac_move_or_jog_abs_queue( mp,
    requested_position, 0);
}

```

5.6.4.40 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 2031 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
char s[512];    // PMAC command string buffer
double u2c;

pthread_mutex_lock( &(mp->mutex));

u2c = lsredis_getd( mp->u2c);
if( u2c == 0.0 || time <= 0.0) {
    //
    // Shouldn't try moving a motor that has no units defined
    //
    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 / mp->max_accel;
q100 = 1 << (mp->coord_num - 1);
pthread_mutex_unlock( &(mp->mutex));

snprintf( s, sizeof(s)-1, "%d Q10=%d Q11=%d Q12=%d Q13=%d Q100=%d B240R",
    coord_num, q10, q11, q12, q13, q100);
pthread_mutex_lock( &(mp->mutex));
mp->pq = lspmac_SockSendline_nr( s);
pthread_mutex_unlock( &(mp->mutex));
}

```

5.6.4.41 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 2320 of file lspmac.c.

```

{
    struct timespec wt;
    int return_code;
    pmac_cmd_queue_t *pq;

    //
    // Copy the queue item for the most recent move request
    //
    pthread_mutex_lock( &(mp->mutex));
    pq = mp->pq;
    pthread_mutex_unlock( &(mp->mutex));

    pthread_mutex_lock( &pmac_queue_mutex);
    //
    // wait for the command to be sent
    //
    while( pq->time_sent.tv_sec==0)
        pthread_cond_wait( &pmac_queue_cond, &pmac_queue_mutex
        );
}

```

```

//
// set the timeout to be long enough after we sent the motion request to
// ensure that
// we will have read back the motor moving status but not so long that the
// timeout causes
// problems;
//
wt.tv_sec = pq->time_sent.tv_sec;
wt.tv_nsec = pq->time_sent.tv_nsec + 500000000;

pthread_mutex_unlock( &pmac_queue_mutex);

if( wt.tv_nsec >= 1000000000) {
    wt.tv_nsec -= 1000000000;
    wt.tv_sec += 1;
}

//
// wait for the motion to have started
// This will time out if the motion ends before we can read the status back
// hence the added complication of time stamp of the sent packet.
//

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));
}

```

5.6.4.42 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 1845 of file lspmac.c.

```

{
    char s[512];
    double y;
    double u2c;

    pthread_mutex_lock( &(mp->mutex));

    u2c = lsredis_getd( mp->u2c);
    mp->requested_position = requested_position;

    if( mp->nlut > 0 && mp->lut != NULL) {
        mp->requested_pos_cnts = u2c * lspmac_lut( mp->
            nlut, mp->lut, requested_position);
        mp->not_done = 1;
        mp->motion_seen = 0;

        //
        // By convention requested_pos_cnts scales from 0 to 100
        // for the lights u2c converts this to 0 to 16,000
    }
}

```

```

    // for the scintillator focus this is 0 to 32,000
    //
    snprintf( s, sizeof(s)-1, "%s=%d", mp->dac_mvar, mp->
        requested_pos_cnts);
    mp->pq = lspmac_SockSendline_nr( s);
}

pthread_mutex_unlock( &(mp->mutex));
}

```

5.6.4.43 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 1880 of file lspmac.c.

```

{
    char s[512];
    double y;
    pthread_mutex_lock( &(mp->mutex));

    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;

        snprintf( s, sizeof(s)-1, "##%d j=%d", mp->motor_num, mp->
            requested_pos_cnts);
        mp->pq = lspmac_SockSendline_nr( s);
    }

    pthread_mutex_unlock( &(mp->mutex));
}

```

5.6.4.44 void lspmac_newKV_cb (char * event)

Definition at line 2749 of file lspmac.c.

```

{
    lspmac_motor_t    *d;
    lskvs_kvs_t        *p;
    lskvs_kvs_list_t   *q;
    lskvs_kvs_list_t   *r;
    int i;

    pthread_rwlock_rdlock( &lskvs_rwlock);
    p = lskvs_kvs;
    pthread_rwlock_unlock( &lskvs_rwlock);

    while( p != NULL) {
        for( i=0; i<lspmac_nmotors; i++) {
            d = &(lspmac_motors[i]);

            if( regexec( &(d->preset_regex), p->k, 0, NULL, 0) == 0) {
                for( q = d->presets; q != NULL; q = q->next)
                    if( strcmp( q->kvs->k, p->k) == 0)
                        break;
                if( q == NULL) {
                    //
                    // We don't know about this preset yet. Add it to our list.

```

```

    //
    r = calloc( 1, sizeof( *r));
    if( r == NULL) {
        lslogging_log_message( "lspmac_newKV_cb: Out
of memory for kv %s", p->k);
        exit( -1);
    }
    r->kvs = p;
    pthread_mutex_lock( &(d->mutex));
    r->next = d->presets;
    d->presets = r;
    pthread_mutex_unlock( &(d->mutex));
    lslogging_log_message( "lspmac_newKV_cb: added
'%s' with value '%s' to motor '%s'", p->k, p->v, d->name);
}
}
p = p->next;
}
}

```

5.6.4.45 void lspmac_next_state ()

State machine logic.

Given the current state, generate the next one

Definition at line 1704 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 &&
    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 state 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 state 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;
}
}

```

5.6.4.46 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 1241 of file lspmac.c.

```

{
char s[512], *sp;
int homing1, homing2;
double u2c;

pthread_mutex_lock( &(mp->mutex));

//
// if this time and last time were both "in position"
// and the position changed significantly then log the event
//
// On E omega has been observed to change by 0x10000 on its own

```

```

// with no real motion.
//
if( mp->status2 & 1 && mp->status2 == *mp->status2_p
    && abs( mp->actual_pos_cnts - *mp->actual_pos_cnts_p
    ) > 256) {
    // lslogging_log_message( "Instantaneous change: %s old status1: %0x,
    // new status1: %0x, old status2: %0x, new status2: %0x, old cnts: %0x, new cnts:
    // %0x",
    // mp->name, mp->status1, *mp->status1_p, mp->status2,
    // *mp->status2_p, mp->actual_pos_cnts, *mp->actual_pos_cnts_p);

    //
    // At this point we'll just log the event and return
    // There is no reason to believe the change is real.
    //
    // There is a non-zero probability that the first value is the bad one and
    // any value afterwards will be taken as
    // wrong. Homing (or moving) the motor should fix this. There is a
    // non-zero probability 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");
}

//
// maybe look for omega zero crossing
//
if( mp->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_motor_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
//
mp->status1 = *mp->status1_p;
mp->status2 = *mp->status2_p;
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, " ");

u2c = lsredis_getd( mp->u2c);

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;
    }
}
snprintf( s, sizeof(s)-1, mp->format, 8, mp->position);

// 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);

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->status2 & 0x000001)
    sp = "In Position";

mvwprintw( mp->win, 6, 1, "%s", LS_DISPLAY_WINDOW_WIDTH
-2, sp);
wnoutrefresh( mp->win);

strncpy( mp->statuss, sp, sizeof( mp->statuss)-1);
mp->statuss[sizeof(mp->statuss)-1] = 0;

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;
}

```

5.6.4.47 `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 554 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;
}

```

5.6.4.48 `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 574 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;
}
```

5.6.4.49 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 530 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;
}
```

5.6.4.50 void lspmac_Reset()

Clear the queue and put the PMAC into a known state.

Definition at line 651 of file lspmac.c.

```

{
    ls_pmac_state = LS_PMAC_STATE_IDLE;

    // clear queue
    ethCmdReply = ethCmdOn;
    ethCmdOff = ethCmdOn;

    lspmac_SockFlush();
}
```

5.6.4.51 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 366 of file lspmac.c.

```

{
    int i, foundone, up;
    double m;
    double y1, y2, x1, x2, x;

    foundone = 0;
    if( lut != NULL && nlut > 1) {
```

```

if( lut[1] < lut[2*nlut-1])
    up = 1;
else
    up = 0;

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];
    }
    if( i==0 && ( up ? y1 > y : y1 < y)) {
        x = x1;
        foundone = 1;
        break;
    }
    if( y1 == y) {
        x = x1;
        foundone = 1;
        break;
    }
    if( ( i < 2*nlut-2) && (up ? y < y2 : y > y2)) {
        m = (x2 - x1) / (y2 - y1);
        x = m * (y - y1) + x1;
        foundone = 1;
        break;
    }
}
if( foundone == 0 ) {
    x = lut[2*(nlut-1)];
}
return x;
}
return 0.0;
}

```

5.6.4.52 void lspmac_run ()

Start up the lspmac thread.

Definition at line 2792 of file lspmac.c.

```

{
pthread_create( &pmac_thread, NULL, lspmac_worker,
    NULL);
lsevents_add_listener( "NewKV", lspmac_newKV_cb
    );
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);
}

```

5.6.4.53 void lspmac_scint_dried_cb (char * event)

Turn off the dryer.

Parameters

<i>event</i>	required by protocol
--------------	----------------------

Definition at line 2743 of file lspmac.c.

```

{
lslogging_log_message( "lspmac_scint_dried_cb: Stopping

```

```

        dryer");
    dryer->moveAbs( dryer, 0.0);
}

```

5.6.4.54 void lspmac_scint_inPosition_cb (char * event)

Maybe start drying off the scintillator.

Parameters

<i>event</i>	required by protocol
--------------	----------------------

Definition at line 2682 of file lspmac.c.

```

{
    double pos;
    double cover;
    int err;

    pthread_mutex_lock( &(scint->mutex));
    pos = scint->position;
    cover = lskvs_find_preset_position( scint, "
        Cover", &err);
    pthread_mutex_unlock( &(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);
    }
}

```

5.6.4.55 pmac_cmd_queue_t* lspmac_send_command (int rqType, int rq, int wValue, int wIndex, int wLength, unsigned char * data, void(*) (pmac_cmd_queue_t *, int, unsigned char *) responseCB, int no_reply)

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

Definition at line 592 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;

//
// 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);
}

```

5.6.4.56 void lspmac_sendcmd (void(*) (pmac_cmd_queue_t *, int, unsigned char *) responseCB, char * fmt, ...)

PMAC command with call back.

Parameters

in	<i>responseCB</i>	our callback routine
in	<i>fmt</i>	printf style format string

Definition at line 1684 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_SENDBLINE, 0, 0, strlen(tmps), tmps, responseCB, 0);
}

```

5.6.4.57 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 1665 of file lspmac.c.

```

{
    static char tmps[1024];
    va_list arg_ptr;

```



```

va_start( arg_ptr, fmt);
vsprintf( 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);
}

```

5.6.4.58 void lspmac.SendControlReplyPrintCB (pmac_cmd_queue_t * cmd, int nreceived, unsigned 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 910 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, buff);
pthread_mutex_lock( &ncurses_mutex);
wnoutrefresh( term_output);
wnoutrefresh( term_input);
doupdate();
pthread_mutex_unlock( &ncurses_mutex);
}

```

5.6.4.59 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	<i>evt</i>	pollfd object returned by poll
----	------------	--------------------------------

Definition at line 698 of file lspmac.c.

```

{
static unsigned 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) {
            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) {
        unsigned char *newbuff;

        receiveBufferSize += 1400;
        newbuff = calloc( receiveBufferSize, sizeof( unsigned char));
        if( newbuff == NULL) {
            lslogging_log_message( "Out of memory");
            exit( -1);
        }
        memcpy( newbuff, receiveBuffer, receiveBufferIn);
        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;
}
}
}

```

5.6.4.60 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 1069 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;
    }

    // Not sure what kind of status makes sense to report
    mp->statuss[0] = 0;

    pthread_mutex_unlock( &lspmac_shutter_mutex);
}

```

5.6.4.61 void lspmac.SockFlush ()

Reset the PMAC socket from the PMAC side.

Puts the PMAC into a known communications state

Definition at line 644 of file lspmac.c.

```

{
    lspmac_send_command( VR_DOWNLOAD, VR_PMAC_FLUSH
        , 0, 0, 0, NULL, NULL, 1);
}

```

5.6.4.62 pmac_cmd_queue_t* lspmac.SockGetmem (int offset, int nbytes)

Request a chunk of memory to be returned.

Not currently used

Parameters

in	<i>offset</i>	Offset in PMAC Double Buffer
in	<i>nbytes</i>	Number of bytes to request

Definition at line 947 of file lspmac.c.

```

{

```

```

return lspmac_send_command( VR_UPLOAD,
    VR_PMAC_GETMEM, offset, 0, nbytes, NULL, lspmac_GetmemReplyCB
    , 0);
}

```

5.6.4.63 pmac_cmd_queue_t* lspmac_SockSendControlCharPrint (char c)

Send a control character.

Parameters

<i>c</i>	The control character to send
----------	-------------------------------

Definition at line 995 of file lspmac.c.

```

{
return lspmac_send_command( VR_DOWNLOAD,
    VR_PMAC_SENDCTRLCHAR, c, 0, 0, NULL,
    lspmac_SendControlReplyPrintCB, 0);
}

```

5.6.4.64 pmac_cmd_queue_t* lspmac_SockSendline (char * *fmt*, ...)

Send a one line command.

Uses printf style arguments.

Parameters

<i>in</i>	<i>fmt</i>	Printf style format string
-----------	------------	----------------------------

Definition at line 957 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);
}

```

5.6.4.65 pmac_cmd_queue_t* lspmac_SockSendline_nr (char * *fmt*, ...)

Send a command and ignore the response.

Parameters

<i>in</i>	<i>fmt</i>	Printf style format string
-----------	------------	----------------------------

Definition at line 976 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_SENDFLINE, 0, 0, strlen( s), s, NULL, 1);
}

```

5.6.4.66 `lspmac_motor_t* lspmac_soft_motor_init(lspmac_motor_t * d, char * name, double scale, void(*) (lspmac_motor_t *, double) moveAbs)`

Definition at line 2539 of file `lspmac.c`.

```

{

    lspmac_nmotors++;
    lskvs_regcomp( &(d->preset_regex), REG_EXTENDED,
        LSPMAC_PRESET_REGEX, name);
    d->presets = NULL;
    d->name = strdup(name);
    d->moveAbs = moveAbs;
    d->read = lspmac_soft_motor_read;
    d->u2c = lsredis_get_obj( "%s.u2c", name);
    d->lut = NULL;
    d->nlut = 0;
    d->actual_pos_cnts_p = calloc( sizeof(int), 1);
    *d->actual_pos_cnts_p = 0;
    d->status1_p = NULL;
    d->status2_p = NULL;
    d->motor_num = -1;
    d->dac_mvar = NULL;
    d->win = NULL;
    d->homing = 0;
    d->lspg_initialized = 0;
}

```

5.6.4.67 `void lspmac_soft_motor_read(lspmac_motor_t * p)`

Dummy routine to read a soft motor.

Definition at line 2534 of file `lspmac.c`.

```

{
}

```

5.6.4.68 `void lspmac_video.rotate(double secs)`

Special motion program to collect centering video.

Definition at line 2133 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

omega->pq = lspmac_SockSendline_nr( "&1 Q10=%.1f
    Q11=%.1f Q12=%.1f Q13=(I117) Q14=(I116) B240R", q10, q11, q12);
pthread_mutex_unlock( &(omega->mutex));
}

```

5.6.4.69 void* lspmac_worker (void * dummy)

Our lspmac worker thread.

Parameters

in	<i>dummy</i>	Unused but required by pthread library
----	--------------	--

Definition at line 1811 of file lspmac.c.

```

{

while( 1) {
    int pollrtn;

    lspmac_next_state();

    if( pmacfd.fd == -1) {
        sleep( 10); // The pmac is not connected. Should we warn someone?
        //
        // This just puts us into a holding pattern until the pmac becomes
        // connected again
        //
        // TODO:
        // Check PMAC initialization logic and our queues to ensure that it is
        // sane to
        // re-initialize things. Probably bad things will happen.
        //
        continue;
    }

    pollrtn = poll( &pmacfd, 1, 10);
    if( pollrtn) {
        lspmac_Service( &pmacfd);
    }
}
}

```

5.6.5 Variable Documentation

5.6.5.1 lspmac_motor_t* alignx

Alignment stage X.

Definition at line 81 of file lspmac.c.

5.6.5.2 lspmac_motor_t* aligny

Alignment stage Y.

Definition at line 82 of file lspmac.c.

5.6.5.3 lspmac_motor_t* alignz

Alignment stage X.

Definition at line 83 of file lspmac.c.

5.6.5.4 **lspmac_motor_t* anal**

Polaroid analyzer motor.

Definition at line 84 of file lspmac.c.

5.6.5.5 **lspmac_motor_t* apery**

Aperture Y.

Definition at line 86 of file lspmac.c.

5.6.5.6 **lspmac_motor_t* aperz**

Aperture Z.

Definition at line 87 of file lspmac.c.

5.6.5.7 **lspmac_motor_t* blight**

Back Light DAC.

Definition at line 98 of file lspmac.c.

5.6.5.8 **lspmac_motor_t* blight_f**

Back light scale factor.

Definition at line 103 of file lspmac.c.

5.6.5.9 **lspmac_motor_t* blight_ud**

Back light Up/Down actuator.

Definition at line 101 of file lspmac.c.

5.6.5.10 **lspmac_motor_t* capy**

Capillary Y.

Definition at line 88 of file lspmac.c.

5.6.5.11 **lspmac_motor_t* capz**

Capillary Z.

Definition at line 89 of file lspmac.c.

5.6.5.12 **lspmac_motor_t* cenx**

Centering Table X.

Definition at line 91 of file lspmac.c.

5.6.5.13 `lspmac_motor_t*` `cen`

Centering Table Y.

Definition at line 92 of file `lspmac.c`.

5.6.5.14 `pmac_cmd_t` `cr_cmd` `[static]`

commands to send out "readready", "getbuffer", controlresponse (initialized in main)

Definition at line 157 of file `lspmac.c`.

5.6.5.15 `lspmac_motor_t*` `cryo`

Move the cryostream towards or away from the crystal.

Definition at line 105 of file `lspmac.c`.

5.6.5.16 `lspmac_bi_t*` `cryo_switch`

that little toggle switch for the cryo

Definition at line 109 of file `lspmac.c`.

5.6.5.17 `unsigned char` `dbmem[64 * 1024]` `[static]`

double buffered memory

Definition at line 147 of file `lspmac.c`.

5.6.5.18 `int` `dbmemIn = 0` `[static]`

next location

Definition at line 148 of file `lspmac.c`.

5.6.5.19 `lspmac_motor_t*` `dryer`

blow air on the scintillator to dry it off

Definition at line 106 of file `lspmac.c`.

5.6.5.20 `unsigned int` `ethCmdOff = 0` `[static]`

points to current command (or none if == `ethCmdOn`)

Definition at line 160 of file `lspmac.c`.

5.6.5.21 `unsigned int` `ethCmdOn = 0` `[static]`

points to next empty PMAC command queue position

Definition at line 159 of file `lspmac.c`.

5.6.5.22 pmac_cmd_queue_t ethCmdQueue[PMAC_CMD_QUEUE_LENGTH] [static]

PMAC command queue.

Definition at line 158 of file lspmac.c.

5.6.5.23 unsigned int ethCmdReply = 0 [static]

Used like ethCmdOff only to deal with the pmac reply to a command.

Definition at line 161 of file lspmac.c.

5.6.5.24 lspmac_motor_t* flight

Front Light DAC.

Definition at line 97 of file lspmac.c.

5.6.5.25 lspmac_motor_t* flight_f

Front light scale factor.

Definition at line 104 of file lspmac.c.

5.6.5.26 lspmac_motor_t* flight_oo

Turn front light on/off.

Definition at line 102 of file lspmac.c.

5.6.5.27 lspmac_motor_t* fluo

Move the fluorescence detector in/out.

Definition at line 107 of file lspmac.c.

5.6.5.28 lspmac_motor_t* fscint

Scintillator Piezo DAC.

Definition at line 99 of file lspmac.c.

5.6.5.29 lspmac_motor_t* fshut

Fast shutter.

Definition at line 96 of file lspmac.c.

5.6.5.30 pmac_cmd_t gb.cmd [static]

Definition at line 157 of file lspmac.c.

5.6.5.31 int getivars = 0 [static]

flag set at initialization to send i vars to db

Definition at line 72 of file lspmac.c.

5.6.5.32 int getmvars = 0 [static]

flag set at initialization to send m vars to db

Definition at line 73 of file lspmac.c.

5.6.5.33 lspmac_motor_t* kappa

Kappa.

Definition at line 93 of file lspmac.c.

5.6.5.34 int linesReceived = 0 [static]

current number of lines received

Definition at line 146 of file lspmac.c.

5.6.5.35 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.

5.6.5.36 lspmac_bi_t lspmac_bis[16]

array of binary inputs

Definition at line 75 of file lspmac.c.

5.6.5.37 lspmac_motor_t lspmac_motors[48]

All our motors.

Definition at line 78 of file lspmac.c.

5.6.5.38 pthread_cond_t lspmac_moving_cond

Wait for motor(s) to finish moving condition.

Definition at line 58 of file lspmac.c.

5.6.5.39 int lspmac_moving_flags

Flag used to implement motor moving condition.

Definition at line 59 of file lspmac.c.

5.6.5.40 pthread_mutex_t lspmac_moving_mutex

Coordinate moving motors between threads.

Definition at line 57 of file lspmac.c.

5.6.5.41 int lspmac_nbis = 0

number of active binary inputs

Definition at line 76 of file lspmac.c.

5.6.5.42 int lspmac_nmotors = 0

The number of motors we manage.

Definition at line 79 of file lspmac.c.

5.6.5.43 pthread_cond_t lspmac_shutter_cond

Allows waiting for the shutter status to change.

Definition at line 56 of file lspmac.c.

5.6.5.44 int lspmac_shutter_has_opened

Indicates that the shutter had opened, perhaps briefly even if the state did not change.

Definition at line 54 of file lspmac.c.

5.6.5.45 pthread_mutex_t lspmac_shutter_mutex

Coordinates threads reading shutter status.

Definition at line 55 of file lspmac.c.

5.6.5.46 int lspmac_shutter_state

State of the shutter, used to detect changes.

Definition at line 53 of file lspmac.c.

5.6.5.47 struct timespec lspmac_status_last_time [static]

Time the status was read.

Definition at line 65 of file lspmac.c.

5.6.5.48 struct timespec lspmac_status_time [static]

Time the status was read.

Definition at line 64 of file lspmac.c.

5.6.5.49 md2_status_t md2_status [static]

Buffer for MD2 Status.

Definition at line 295 of file lspmac.c.

5.6.5.50 pthread_mutex_t md2_status_mutex

Synchronize reading/writing status buffer.

Definition at line 296 of file lspmac.c.

5.6.5.51 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 153 of file lspmac.c.

5.6.5.52 lspmac_motor_t* omega

MD2 omega axis (the air bearing)

Definition at line 80 of file lspmac.c.

5.6.5.53 int omega_zero_search = 0 [static]

Indicate we'd really like to know when omega crosses zero.

Definition at line 61 of file lspmac.c.

5.6.5.54 struct timespec omega_zero_time

Time we believe that omega crossed zero.

Definition at line 63 of file lspmac.c.

5.6.5.55 double omega_zero_velocity = 0 [static]

rate (cnts/sec) that omega was traveling when it crossed zero

Definition at line 62 of file lspmac.c.

5.6.5.56 lspmac_motor_t* phi

Phi (not data collection axis)

Definition at line 94 of file lspmac.c.

5.6.5.57 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"
}

```

Decode the errors perhaps returned by the PMAC.

Definition at line 164 of file `lspmac.c`.

5.6.5.58 `pthread_cond_t pmac_queue_cond`

wait for a command to be sent to PMAC before continuing

Definition at line 69 of file `lspmac.c`.

5.6.5.59 `pthread_mutex_t pmac_queue_mutex`

manage access to the pmac command queue

Definition at line 68 of file `lspmac.c`.

5.6.5.60 `pthread_t pmac_thread` `[static]`

our thread to manage access and communication to the pmac

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

5.6.5.61 `struct pollfd pmacfd` `[static]`

our poll structure

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

5.6.5.62 `pmac_cmd_t rr_cmd` `[static]`

Definition at line 157 of file `lspmac.c`.

5.6.5.63 `lspmac_motor_t* scint`

Scintillator Z.

Definition at line 90 of file `lspmac.c`.

5.6.5.64 `lspmac_motor_t* zoom`

Optical zoom.

Definition at line 85 of file `lspmac.c`.

5.7 Lsredis.c File Reference

Support redis hash synchronization.

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

Functions

- int [lsredis_isvalid](#) (lsredis_obj_t *p)
- void [lsredis_set_invalid](#) (lsredis_obj_t *p)
- 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.
- 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)
- 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.
- [lsredis_obj_t * lsredis_get_obj](#) (char *fmt,...)
- void [redisDisconnectCB](#) (const redisAsyncContext *ac, int status)
call back incase a redis server becomes disconnected TODO: reconnect
- void [lsredis_addRead](#) (void *data)
hook to manage 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_debugCB](#) (redisAsyncContext *ac, void *reply, void *privdata)
Log the reply.
- void [lsredis_subCB](#) (redisAsyncContext *ac, void *reply, void *privdata)
Use the publication to request the new value.
- void [lsredis_maybe_add_key](#) (char *k)
- void [lsredis_keysCB](#) (redisAsyncContext *ac, void *reply, void *privdata)
Sift through the keys to find ones we like.
- void [lsredis_select](#) (char *re)
set regexp to select variables we are interested in following
- 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_worker](#) (void *dummy)
subscribe to changes and service sockets
- void [lsredis_run](#) ()

Variables

- static pthread_t [lsredis_thread](#)
- static [lsredis_obj_t](#) * [lsredis_objs](#) = NULL
- static pthread_mutex_t [lsredis_objs_mutex](#)
- static pthread_mutex_t [lsredis_ro_mutex](#)
keep from having more than one thread send a rediscommand to the read/only server
- static pthread_mutex_t [lsredis_wr_mutex](#)
keep from having more than one thread send a rediscommand to the write/read server
- 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](#)

5.7.1 Detailed Description

Support redis hash synchronization.

Date

2012

Author

Keith Brister

Copyright

All Rights Reserved

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

5.7.2 Function Documentation

5.7.2.1 [lsredis_obj_t](#)* [lsredis_get_obj](#)(char * *key*)

Maybe add a new object Used internally for this module.

Definition at line 217 of file [lsredis.c](#).

```

{
    lsredis\_obj\_t *p;
    regmatch_t pmatch[2];
    int err;
    char *name;

    // 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;
}

// printf( "_lsredis_get_obj: received key '%s'", key);
// fflush( stdout);

pthread_mutex_lock( &lsredis_objs_mutex);
// If the key is already there then just return it
//
for( p = lsredis_objs; p != NULL; p = p->next) {
    if( strcmp( key, p->key) == 0) {
        break;
    }
}

if( p != NULL) {
    pthread_mutex_unlock( &lsredis_objs_mutex);
    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 (evetns_name)");
        exit( -1);
    }

    pthread_mutex_init( &p->mutex, NULL);
    pthread_cond_init( &p->cond, NULL);
    p->valid = 0;
    lsevents_send_event( "%s Invalid", p->events_name
        );
    p->wait_for_me = 0;
    p->key = strdup( key);
    p->getstr = lsredis_getstr;
    p->getd = lsredis_getd;
    p->getl = lsredis_getl;
    p->hits = 0;

    p->next = lsredis_objs;
    lsredis_objs = p;

    pthread_mutex_unlock( &lsredis_objs_mutex);

    lslogging_log_message( "_lsredis_get_obj: added %s",
        key);
}
//
// We arrive here with the valid flag lowered. Go ahead and request the
// latest value.
//
pthread_mutex_lock( &lsredis_ro_mutex);
redisAsyncCommand( roac, lsredis_hgetCB, p, "HGET %s VALUE"
    , key);
pthread_mutex_unlock( &lsredis_ro_mutex);

return p;
}

```

5.7.2.2 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 48 of file lsredis.c.

```
{
```

```

if( strlen(v) >= 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);
    }
}
strcpy( p->value, v);
p->value[p->value_length-1] = 0;

p->valid = 1;
lsevents_send_event( "%s Valid", p->events_name
    );
}

```

5.7.2.3 void lsredis_addRead (void * data)

hook to mange read events

Definition at line 334 of file lsredis.c.

```

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

```

5.7.2.4 void lsredis_addWrite (void * data)

hook to manage write events

Definition at line 350 of file lsredis.c.

```

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

```

5.7.2.5 void lsredis_cleanup (void * data)

hook to clean up TODO: figure out what we are supposed to do here and do it

Definition at line 367 of file lsredis.c.

```

{
    struct pollfd *pfd;
    pfd = (struct pollfd *)data;
    pfd->events &= ~(POLLOUT | POLLIN);
    pfd->fd = -1;
}

```

5.7.2.6 void lsredis_debugCB (redisAsyncContext * ac, void * reply, void * privdata)

Log the reply.

Definition at line 377 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<r->elements; i++)
        lsredis_debugCB( ac, r->element[i], NULL);
    indentlevel--;
    break;

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

```

5.7.2.7 void lsredis_delRead (void * *data*)

hook to manage "don't need to read" events

Definition at line 342 of file lsredis.c.

```

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

```

5.7.2.8 void lsredis_delWrite (void * *data*)

hook to manage "don't need to write anymore" events

Definition at line 358 of file lsredis.c.

```

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

```

5.7.2.9 void lsredis_fd_service (struct pollfd * evt)

service the socket requests

Definition at line 658 of file lsredis.c.

```

{
    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);
    }
}

```

5.7.2.10 lsredis_obj_t* lsredis_get_obj(char * fmt, ...)

Definition at line 297 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;
    rtn = _lsredis_get_obj( kp);
    free( kp);
    return rtn;
}

```

5.7.2.11 double lsredis_getd(lsredis_obj_t * p)

Definition at line 160 of file lsredis.c.

```

{
    double rtn;

    pthread_mutex_lock( &p->mutex);
    while( p->valid == 0)
        pthread_cond_wait( &p->cond, &p->mutex);

    errno = 0;
    rtn = strtod( p->value, NULL);
    pthread_mutex_unlock( &p->mutex);

    if( errno != 0)

```

```

    lslogging_log_message( "lsredis_getd: %s", strerror);
return rtn;
}

```

5.7.2.12 long int lsredis_getl (lsredis_obj_t * p)

Definition at line 177 of file Isredis.c.

```

{
    long int rtn;

    pthread_mutex_lock( &p->mutex);
    while( p->valid == 0)
        pthread_cond_wait( &p->cond, &p->mutex);

    errno = 0;
    rtn = strtol( p->value, NULL, 10);
    pthread_mutex_unlock( &p->mutex);

    if( errno != 0)
        lslogging_log_message( "lsredis_getd: %s", strerror);

    return rtn;
}

```

5.7.2.13 char* lsredis_getstr (lsredis_obj_t * p)

return a copy of the key's string value

Definition at line 92 of file Isredis.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;
}

```

5.7.2.14 void lsredis_hgetCB (redisAsyncContext * ac, void * reply, void * privdata)

Definition at line 194 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?");

    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);
    }
}

```

5.7.2.15 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 601 of file lsredis.c.

```

{

    lsredis_head = strdup( head);
    lsredis_publisher = strdup( pub);

    pthread_mutex_init( &lsredis_objs_mutex, NULL);
    pthread_mutex_init( &lsredis_ro_mutex, NULL);
    pthread_mutex_init( &lsredis_wr_mutex, 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);
    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;

    lsredis_select( re);
}

```

5.7.2.16 int lsredis_isvalid (lsredis_obj_t * *p*)

Definition at line 28 of file lsredis.c.

```

{

    int rtn;

    pthread_mutex_lock( &p->mutex);
    rtn = p->valid;

```

```
pthread_mutex_unlock( &p->mutex);
return rtn;
}
```

5.7.2.17 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 546 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<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);
    }
}
}
```

5.7.2.18 void Lsredis.maybe_add_key (char * k)

Definition at line 538 of file Lsredis.c.

```
{
if( regexec( &lsredis_key_select_regex, k, 0, NULL, 0
    ) == 0) {
    _lsredis_get_obj( k);
}
}
```

5.7.2.19 void Lsredis.run ()

Definition at line 731 of file Lsredis.c.

```
{
pthread_create( &lsredis_thread, NULL, lsredis_worker
    , NULL);
}
```

5.7.2.20 void Lsredis.select (char * re)

set regexp to select variables we are interested in following

Note that redis only supports glob matching while we'd prefer something a tad more useful. See <http://xkcd.com/208>

Definition at line 575 of file Lsredis.c.

```

{
    int err;
    char *errmsg;
    int nerrmsg;

    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);
        }
    }

    pthread_mutex_lock( &lsredis_ro_mutex);
    redisAsyncCommand( roac, lsredis_keysCB, NULL, "KEYS *");
    pthread_mutex_unlock( &lsredis_ro_mutex);
}

```

5.7.2.21 void lsredis.set_invalid(lsredis_obj_t * p)

Definition at line 38 of file lsredis.c.

```

{
    pthread_mutex_lock( &p->mutex);
    p->valid = 0;
    lsevents_send_event( "%s Invalid", p->events_name
        );
    pthread_mutex_unlock( &p->mutex);
}

```

5.7.2.22 void lsredis.set_value(lsredis_obj_t * p, char * fmt, ...)

Set the value of a redis object and make it valid.

Called by mgetCB to set the value as it is in redis Maybe TODO: we've arbitrarily set the maximum size of a value here. Although I cannot imagine needed bigger values it would not be a big deal to enable it.

Definition at line 72 of file lsredis.c.

```

{
    va_list arg_ptr;
    char v[512];

    va_start( arg_ptr, fmt);
    vsnprintf( v, sizeof(v)-1, fmt, arg_ptr);
    va_end( arg_ptr);

    v[sizeof(v)-1] = 0;

    pthread_mutex_lock( &p->mutex);

    _lsredis_set_value( p, v);

    pthread_cond_signal( &p->cond);
    pthread_mutex_unlock( &p->mutex);
}

```

5.7.2.23 void lsredis.setstr(lsredis_obj_t * p, char * fmt, ...)

Set the value and update redis.

Note that lsredis.set_value sets the value based on redis while here we set redis based on the value Arbitray maximum string length set here. TODO: Probably this limit should be removed at some point.

redisAsyncCommandArgv used instead of redisAsyncCommand 'cause it's easier (and possible) to deal with strings that would otherwise cause hiredis to emit a bad command, like those containing spaces. < invalidate the current value: set_value will fix this and signal waiting threads

< up the count of times we need to see ourselves published before we start listening to others again

< key is "immutable" (not really a C feature). In any case no one is going to be changing it so it's cool to read it without mutex protection.

< redisAsyncCommandArgv shouldn't need to access this after it's made up it's packet (before it returns) so we should be OK with this location disappearing soon.

Definition at line 116 of file Isredis.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);

    if( p->valid && strcmp( v, p->value) == 0) {
        // nothing to do
        pthread_mutex_unlock( &p->mutex);
        return;
    }

    p->valid = 0;
    lsevents_send_event( "%s Invalid", p->events_name
    );
    p->wait_for_me++;

    argv[0] = "HSET";
    argv[1] = p->key;
    argv[2] = "VALUE";
    argv[3] = v;

    pthread_mutex_lock( &lsredis_wr_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_wr_mutex);

    // Assume redis will take exactly the value we sent it
    //
    _lsredis_set_value( p, v);
    pthread_cond_signal( &p->cond);
    pthread_mutex_unlock( &p->mutex);
}

```

5.7.2.24 void lsredis_subCB (redisAsyncContext * ac, void * reply, void * privdata)

Use the publication to request the new value.

Definition at line 429 of file Isredis.c.

```

{
    redisReply *r;
    lsredis_obj_t *p, *last, *last2;
    char *k;
    char *publisher;

    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
    //
    pthread_mutex_lock( &lsredis_objs_mutex);
    last = NULL;
    last2 = NULL;
    for( p=lsredis_objs; p != NULL; p = p->next) {
        if( strcmp( p->key, k) == 0) {
            p->hits++;
            //
            // Maybe reorder our list so the most often updated objects
            // eventually bump up to the beginning of the list.
            // That "hits+4" keeps us from oscillating when objects are accessed
            // equally
            //
            if( last != NULL && last->hits < p->hits+4) {
                last->next = p->next;
                p->next = last;
                if( last2 != NULL)
                    last2->next = p;
                else
                    lsredis_objs = p;
            }
            break;
        }
        last2 = last;
        last = p;
    }
    pthread_mutex_unlock( &lsredis_objs_mutex);

    if( p == NULL) {
        //
        // Regardless of who the publisher is, apparently there is a key we've
        // not seen before
        //
        _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;
    }
}

```

```

lsredis_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.
//
pthread_mutex_lock( &lsredis_ro_mutex);
redisAsyncCommand( roac, lsredis_hgetCB, p, "HGET %s
    VALUE", k);
pthread_mutex_unlock( &lsredis_ro_mutex);
}
}
}

```

5.7.2.25 void* lsredis_worker (void * dummy)

subscribe to changes and service sockets

< array of pollfd's for the poll function, one entry per connection

< number of active elements in fda

< poll timeout, in millisecs (of course)

Definition at line 682 of file Lsredis.c.

```

{
static struct pollfd fda[3];
static int nfda = 0;
static int poll_timeout_ms = -1;
int pollrtn;
int i;

pthread_mutex_lock( &lsredis_ro_mutex);
if( redisAsyncCommand( subac, lsredis_subCB, NULL, "
    PSUBSCRIBE REDIS_KV_CONNECTOR UI*" ) == REDIS_ERR) {
    lslogging_log_message( "Error sending PSUBSCRIBE
        command");
}
pthread_mutex_unlock( &lsredis_ro_mutex);

while(1) {
    nfda = 0;

    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++;
    }

    pollrtn = poll( fda, nfda, poll_timeout_ms);

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

```

5.7.2.26 void redisDisconnectCB (const redisAsyncContext * *ac*, int *status*)

call back incase a redis server becomes disconnected TODO: reconnect

Definition at line 326 of file lsredis.c.

```

{
    if( status != REDIS_OK) {
        lslogging_log_message( "lsredis: Disconnected with
            status %d", status);
    }
}

```

5.7.3 Variable Documentation

5.7.3.1 char* lsredis_head = NULL [static]

Definition at line 22 of file lsredis.c.

5.7.3.2 regex_t lsredis_key_select_regex [static]

Definition at line 21 of file lsredis.c.

5.7.3.3 lsredis_obj_t* lsredis_objs = NULL [static]

Definition at line 11 of file lsredis.c.

5.7.3.4 pthread_mutex_t lsredis_objs_mutex [static]

Definition at line 12 of file lsredis.c.

5.7.3.5 char* lsredis_publisher = NULL [static]

Definition at line 20 of file lsredis.c.

5.7.3.6 pthread_mutex_t lsredis_ro_mutex [static]

keep from having more than one thread send a rediscommand to the read/only server

Definition at line 13 of file lsredis.c.

5.7.3.7 pthread_t lsredis_thread [static]

Definition at line 9 of file lsredis.c.

5.7.3.8 pthread_mutex_t lsredis_wr_mutex [static]

keep from having more than one thread send a rediscommand to the write/read server

Definition at line 14 of file lsredis.c.

5.7.3.9 redisAsyncContext* roac [static]

Definition at line 17 of file lsredis.c.

5.7.3.10 struct pollfd rofd [static]

Definition at line 25 of file lsredis.c.

5.7.3.11 redisAsyncContext* subac [static]

Definition at line 16 of file lsredis.c.

5.7.3.12 struct pollfd subfd [static]

Definition at line 24 of file lsredis.c.

5.7.3.13 redisAsyncContext* wrac [static]

Definition at line 18 of file lsredis.c.

5.7.3.14 struct pollfd wrfd [static]

Definition at line 26 of file lsredis.c.

5.8 Istimer.c File Reference

Support for delayed and periodic events.

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

Data Structures

- struct [lstimer_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 [lstimer_list_struct](#) [lstimer_list_t](#)
Everything we need to know about a timer.

Functions

- void [lstimer_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 * [lstimer_worker](#) (void *dummy)
- Our worker.*
- void [lstimer_init](#) ()
- Initialize the timer list and pthread stuff.*
- void [lstimer_run](#) ()
- Start up our thread.*

Variables

- static int [lstimer_active_timers](#) = 0
- count of the number timers we are tracking*
- static [lstimer_list_t](#) [lstimer_list](#) [[LSTIMER_LIST_LENGTH](#)]
- Our timer list.*
- static pthread_t [lstimer_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*

5.8.1 Detailed Description

Support for delayed and periodic events.

Date

2012

Author

Keith Brister

Copyright

All Rights Reserved

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

5.8.2 Macro Definition Documentation

5.8.2.1 #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](#).

5.8.2.2 #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`.

5.8.3 Typedef Documentation

5.8.3.1 typedef struct `Lstimer_list_struct` `Lstimer_list_t`

Everything we need to know about a timer.

5.8.4 Function Documentation

5.8.4.1 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);
}

```

5.8.4.2 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;
    }
}

```

```

strcpy( 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);
}

```

5.8.4.3 void lstimer_init ()

Initialize the timer list and pthread stuff.

Definition at line 262 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);
}

```

5.8.4.4 void lstimer_run ()

Start up our thread.

Definition at line 276 of file lstimer.c.

```

{
    pthread_create( &lstimer_thread, NULL, lstimer_worker
, NULL);
}

```

5.8.4.5 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
            i,
            known_timers;

        struct timespec now;

        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("ltimer_worker: sigaction
                                failed");
            exit(-1);
        }

        // Create the timer
        //
        sev.sigev_notify = SIGEV_SIGNAL;
        sev.sigev_signo = SIGRTMIN;
        sev.sigev_value.sival_ptr = &ltimer_timerid;
        timer_create(CLOCK_REALTIME, &sev, &ltimer_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(&ltimer_mutex);

            while( new_timer == 0)
                pthread_cond_wait(&ltimer_cond, &ltimer_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(&ltimer_mutex);

            // Let the signals rain down
            //
            sigprocmask(SIG_UNBLOCK, &mask, NULL);
        }
    }
}

```

5.8.4.6 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;

        lstimer_list_t *p;
        struct timespec now, then, soonest;
        struct itimerspec its;

        //
        // Did I remind you not to let this thread own the lstimer mutex outside of
        // this
        // service routine when SIGRTMIN is active?
        //

        // Call with lstimer_mutex locked

        clock_gettime( CLOCK_REALTIME, &now);
        //
        // Project a tad into the future
        then.tv_sec = now.tv_sec + (now.tv_nsec + LSTIMER_RESOLUTION_NSECS
            ) / 1000000000;
        then.tv_nsec = (now.tv_nsec + LSTIMER_RESOLUTION_NSECS
            ) % 1000000000;

        found_active = 0;
        for( i=0; i<lstimer_active_timers; i++) {
            p = &(lstimer_list[i]);
            if( p->shots != 0) {
                found_active++;
                if( p->next_secs < then.tv_sec || (p->next_secs ==
                    then.tv_sec && p->next_nsecs <= then.tv_nsec)) {
                    lsevents_send_event( p->event);
                    //
                    // After sending the event, compute the next time we need to do this
                    //
                    p->last_secs = now.tv_sec;
                    p->last_nsecs = now.tv_nsec;
                    p->ncalls++;
                    //
                    // Decrement non-infinite loops
                    if( p->shots != -1)
                        p->shots--;
                    if( p->shots == 0) {
                        //
                        // Take this timer out of the mix
                        lstimer_active_timers--;
                    } else {
                        p->next_secs = p->init_secs + (p->ncalls+1)
                        * p->delay_secs + (p->init_nsecs + (p->ncalls+1)*p->
                        delay_nsecs)/1000000000;
                        p->next_nsecs = (p->init_nsecs + (p->ncalls
                        +1)*p->delay_nsecs) % 1000000000;
                    }
                }

                if( found_active == 1) {
                    soonest.tv_sec = p->next_secs;
                    soonest.tv_nsec = p->next_nsecs;
                } else {
                    if( soonest.tv_sec > p->next_secs || (soonest.tv_sec == p->
                    next_secs && soonest.tv_nsec > p->next_nsecs)) {
                        soonest.tv_sec = p->next_secs;
                        soonest.tv_nsec = p->next_nsecs;
                    }
                }
            }
        }

        if( soonest.tv_sec != 0) {
            its.it_value.tv_sec = soonest.tv_sec;
            its.it_value.tv_nsec = soonest.tv_nsec;
            its.it_interval.tv_sec = 0;
            its.it_interval.tv_nsec = 0;
            timer_settime( lstimer_timerid, TIMER_ABSTIME, &its, NULL);
        }
    }
}

```

5.8.5 Variable Documentation

5.8.5.1 `int lstimer_active_timers = 0` [static]

count of the number timers we are tracking

Definition at line 18 of file ltimer.c.

5.8.5.2 pthread_cond_t ltimer_cond [static]

allows us to be idle when there is nothing to do

Definition at line 40 of file ltimer.c.

5.8.5.3 ltimer_list_t ltimer_list[LSTIMER_LIST_LENGTH] [static]

Our timer list.

Definition at line 36 of file ltimer.c.

5.8.5.4 pthread_mutex_t ltimer_mutex [static]

protect the timer list

Definition at line 39 of file ltimer.c.

5.8.5.5 pthread_t ltimer_thread [static]

the timer thread

Definition at line 38 of file ltimer.c.

5.8.5.6 timer_t ltimer_timerid [static]

our real time timer

Definition at line 41 of file ltimer.c.

5.8.5.7 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.

5.9 lsupdate.c File Reference

Brings this MD2 code and the database kvs table into agreement.

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

Functions

- void [lsupdate_updateit](#) ()
Query the motors and perhaps tell the DB about it.
- void * [lsupdate_worker](#) (void *dummy)
Our worker thread.
- void [lsupdate_init](#) ()
Initialize this module.
- void [lsupdate_run](#) ()

run the update routines

Variables

- static pthread_t [lsupdate_thread](#)
our worker thread

5.9.1 Detailed Description

Brings this MD2 code and the database kvs table into agreement.

Date

2012

Author

Keith Brister

Copyright

All Rights Reserved

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

5.9.2 Function Documentation

5.9.2.1 void lsupdate_init ()

Initialize this module.

Definition at line 109 of file lsupdate.c.

```

    {
}

```

5.9.2.2 void lsupdate_run ()

run the update routines

Definition at line 114 of file lsupdate.c.

```

    {
        pthread_create( &lsupdate_thread, NULL, lsupdate_worker
            , NULL);
    }

```

5.9.2.3 void lsupdate_updateit ()

Query the motors and perhaps tell the DB about it.

< support for obsolete (ie, non .position) style

Definition at line 15 of file lsupdate.c.

```

    {
static char s[1024];
static char s1[512];
static char s2[512];
lspmac_motor_t *mp;
int i;
int needComma;
int gotone;

needComma = 0;
gotone = 0;
s[0] = 0;
strcpy(s, "select px.kvupdate('{");

for( i=0; i<lspmac_nmotors; i++) {
    mp = &(lspmac_motors[i]);

    pthread_mutex_lock( &(mp->mutex));
    //
    // Bit 0 of lspg_initialized is 0 if we've not yet initialized the motor
    // values via the DB
    // Bit 1 of lspg_initialized is 0 if we've not yet sent any update for this
    // motor
    //
    // Never update if the database has not initialized the motor values
    // Then, always update if we've not done so yet
    // Then, only update if the current position has changed significantly
    //
    if( ((mp->lspg_initialized & 1) == 0) ||
        ((mp->lspg_initialized & 2) != 0) &&
        (fabs( mp->position - mp->reported_position) <
         mp->update_resolution)
        ) {
        pthread_mutex_unlock( &(mp->mutex));
    } else {

        gotone = 1;
        s1[0]=0;

        snprintf( s1, sizeof(s1)-1, mp->update_format, mp->position
        );
        s1[sizeof(s1)-1] = 0;

        if( mp->name != NULL && *mp->name != 0 ) {
            snprintf( s2, sizeof(s2)-1, "\\\"%s\\\",%.3f", mp->name, mp->position
        );
            s2[sizeof(s2)-1] = 0;
        }

        mp->reported_position = mp->position;
        mp->lspg_initialized |= 2;
        pthread_mutex_unlock( &(mp->mutex));

        if( strlen(s2) + strlen(s1) + strlen(s) + 32 >= sizeof( s)-1 ) {
            // send off update now and reset s
            strcat( s, "}'::text[]");
            lspg_query_push( NULL, s);

            s[0] = 0;
            strcpy( s, "select px.kvupdate('{");
            needComma = 0;
            gotone = 0;
        }

        if( needComma)
            strcat( s, "," );
        else
            needComma=1;

        strcat( s, s1);
        if( mp->name != NULL && *mp->name != 0 ) {
            strcat( s, s2);
        }
    }
}

if( gotone) {
    strcat( s, "}'::text[]");
    lspg_query_push( NULL, s);
}
}

```

5.9.2.4 void* lsupdate_worker (void * dummy)

Our worker thread.

Parameters

in	dummy	Unused argument required by protocol
----	-------	--------------------------------------

Definition at line 94 of file lsupdate.c.

```

    {
static struct timespec naptime;

naptime.tv_sec = 0;
naptime.tv_nsec = 500000000;
while( 1 ) {
    lsupdate_updateit();
    nanosleep( &naptime, NULL);
}
}

```

5.9.3 Variable Documentation

5.9.3.1 pthread_t lsupdate_thread [static]

our worker thread

Definition at line 10 of file lsupdate.c.

5.10 md2cmds.c File Reference

Implements commands to run the md2 diffractometer attached to a PMAC controlled by postgresql.

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

Functions

- void [md2cmds_transfer](#) ()
Transfer a sample TODO: Implement.
- 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_prep](#) ()
Sets up a centering table and alignment table move Ensures that when we issue the move command that we can detect that the move happened.
- double [md2cmds_prep_motion](#) (lspmac_motor_t *mp, double pos)
- void [md2cmds_mvcenter_move](#) (double cx, double cy, double ax, double ay, double az)
Move the centering and alignment tables.
- void [md2cmds_mvcenter_wait](#) ()
Wait for the centering and alignment tables to stop moving.
- void [md2cmds_maybe_done_moving_cb](#) (char *event)
Track how many motors are moving.
- 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_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
- pthread_cond_t [md2cmds_moving_cond](#)
coordinate call and response
- 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;
- static pthread_t [md2cmds_thread](#)
- static int [rotating](#) = 0
flag: when omega is in position after a rotate we want to re-home omega

5.10.1 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](#).

5.10.2 Function Documentation

5.10.2.1 void md2cmds_center ()

Move centering and alignment tables as requested TODO: Implement.

Definition at line 737 of file md2cmds.c.

```

    {
}

```

5.10.2.2 void md2cmds_collect ()

Collect some data.

Definition at line 422 of file md2cmds.c.

```

    {
long long skey;
double p170; // start cnts
double p171; // end cnts
double p173; // omega velocity cnts/msec
double p175; // acceleration time (msec)
double p180; // exposure time (msec)
int center_request;
double u2c;

u2c = lsredis_getd( omega->u2c);

//
// reset shutter has opened flag
//
lspmac_SockSendline( "P3001=0 P3002=0");

while( 1) {
    lspg_nextshot_call();

    //
    // This is where we'd tell the md2 to move the organs into position
    //
    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(
            (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)) {

            center_request = 1;
            md2cmds_mvcenter_prep();
            md2cmds_mvcenter_move( lspg_nextshot.
            cx, lspg_nextshot.cy, lspg_nextshot.ax,
            lspg_nextshot.ay, lspg_nextshot.az);
        }
    }

    if( !lspg_nextshot.dsphi_isnull) {
        lspmac_moveabs_queue( phi, lspg_nextshot
        .dsphi);
    }
}

```



```

if( !lspg_nextshot.dskappa_isnull) {
    lspmac_moveabs_queue( kappa, lspg_nextshot
        .dskappa);
}

//
// Wait for all those motors to stop
//
if( center_request) {
    md2cmds_mvcenter_wait();
}

if( !lspg_nextshot.dsphi_isnull) {
    lspmac_moveabs_wait( phi);
}

if( !lspg_nextshot.dskappa_isnull) {
    lspmac_moveabs_wait( kappa);
}

//
// 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.sstart + lspg_nextshot.dsowidth);
p171 = u2c * lspg_nextshot.dsowidth;
p173 = fabs(p180) < 1.e-4 ? 0.0 : u2c * lspg_nextshot.dsowidth
    / p180;
p175 = p173/omega->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_SockSendline( "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);

lspg_query_push( NULL, "SELECT px.unlock_diffractionmeter()");

lspg_query_push( NULL, "SELECT px.shots_set_state(%lld,
    'Writing')", skey);

//
// reset shutter has opened flag
//
lspmac_SockSendline( "P3001=0");
//
// TODO:
// wait for omega to stop moving then position it for the next frame
//

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_mvcenter_prep();
        md2cmds_mvcenter_move( lspg_nextshot.
        cx, lspg_nextshot.cy, lspg_nextshot.ax,
        lspg_nextshot.ay, lspg_nextshot.az);
        md2cmds_mvcenter_wait();
        lspmac_moveabs_wait( cenx);
        lspmac_moveabs_wait( ceny);
        lspmac_moveabs_wait( alignx);
        lspmac_moveabs_wait( aligny);
        lspmac_moveabs_wait( alignz);
    }
}
}
}

```

5.10.2.3 void md2cmds_init ()

Initialize the md2cmds module.

Definition at line 778 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);
}

```

5.10.2.4 void md2cmds_maybe_done_moving_cb(char * event)

Track how many motors are moving.

Definition at line 394 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 {
    //
    // Shouldn't need this but just in case a move was not finished before
    // we're ready
    // this might take care of the problem
    //
    if( md2cmds_moving_count > 0)
        md2cmds_moving_count--;
}

if( md2cmds_moving_count == 0)
    pthread_cond_signal( &md2cmds_moving_cond);
pthread_mutex_unlock( &md2cmds_moving_mutex);
}

```

5.10.2.5 void md2cmds_maybe_rotate_done_cb (char * event)

Now that we are done with the 360 rotation lets rehome right quick.

Definition at line 716 of file md2cmds.c.

```

{
    if( rotating) {
        rotating = 0;
        lspmac_homed_queue( omega);
    }
}

```

5.10.2.6 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 35 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);
}

```

5.10.2.7 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 330 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_motion( cenx, cx);
cy_cts = md2cmds_prep_motion( ceny, cy);
ax_cts = md2cmds_prep_motion( alignx, ax);
ay_cts = md2cmds_prep_motion( aligny, ay);
az_cts = md2cmds_prep_motion( alignz, az);

lspmac_SockSendline( "%2 Q100=2 Q20=%.1f Q21=%.1f B150R",
    cx_cts, cy_cts);
md2cmds_moving_pq = lspmac_SockSendline(
    "%3 Q100=4 Q30=%.1f Q31=%.1f Q32=%.1f B160R", ax_cts, ay_cts, az_cts);
}
```

5.10.2.8 void md2cmds_mvcenter_prep ()

Sets up a centering table and alignment table move Ensures that when we issue the move command that we can detect that the move happened.

Definition at line 235 of file md2cmds.c.

```
{
    lspmac_cmd_queue_t *pq;
    int flag;

    pthread_mutex_lock( &lspmac_moving_mutex);
    flag = (lspmac_moving_flags & 6) != 0;
    pthread_mutex_unlock( &lspmac_moving_mutex);

    //
    // Only wait for the all clear if it's not all clear already
    // Otherwise we may get confused
    //
    if( flag) {
        //
        // Clears the motion flags for coordinate systems 2 and 3
        // Then sets 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
        //

        //
        // Clear the centering and alignment stage flags
        //
        pq = lspmac_SockSendline( "M5075=(M5075 | 6) ^ 6");

        pthread_mutex_lock( &pmac_queue_mutex);
        //
        // wait for the command to be sent
        //
        while( pq->time_sent.tv_sec==0)
            pthread_cond_wait( &pmac_queue_cond, &pmac_queue_mutex
        );
        pthread_mutex_unlock( &pmac_queue_mutex);

        //
        // Make sure the command propagates back to the status
        //
        pthread_mutex_lock( &lspmac_moving_mutex);
        while( (lspmac_moving_flags & 6) != 0)
            pthread_cond_wait( &lspmac_moving_cond, &
                lspmac_moving_mutex);

        lslogging_log_message( "md2cmds_mvcenter_prep:
            lspmac_moving_flags = %d", lspmac_moving_flags);
        pthread_mutex_unlock( &lspmac_moving_mutex);
    }

    //
    // set a flag so the event listener doesn't send a callback too soon
    //
    pthread_mutex_lock( &md2cmds_moving_mutex);
    md2cmds_moving_count = -1;
    pthread_mutex_unlock( &md2cmds_moving_mutex);
}
```

```

//
// Now set the centering and alignment stage flags
//
pq = lspmac_SockSendline( "M5075=(M5075 | 6)");

pthread_mutex_lock( &pmac_queue_mutex);
//
// wait for the command to be sent
//
while( pq->time_sent.tv_sec==0)
    pthread_cond_wait( &pmac_queue_cond, &pmac_queue_mutex
);
pthread_mutex_unlock( &pmac_queue_mutex);

//
// Make sure it propagates
//
pthread_mutex_lock( &lspmac_moving_mutex);
while( (lspmac_moving_flags & 6) != 6)
    pthread_cond_wait( &lspmac_moving_cond, &
        lspmac_moving_mutex);

lslogging_log_message( "md2cmds_mvcenter_prep:
    lspmac_moving_flags = %d", lspmac_moving_flags);
pthread_mutex_unlock( &lspmac_moving_mutex);
}

```

5.10.2.9 void md2cmds_mvcenter_wait ()

Wait for the centering and alignment tables to stop moving.

Definition at line 358 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" callback was received and acted on
// 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
//
while( md2cmds_moving_pq->time_sent.tv_sec==0)
    pthread_cond_wait( &pmac_queue_cond, &pmac_queue_mutex
);
pthread_mutex_unlock( &pmac_queue_mutex);

pthread_mutex_lock( &lspmac_moving_mutex);
while( lspmac_moving_flags & 6)
    pthread_cond_wait( &lspmac_moving_cond, &
        lspmac_moving_mutex);
pthread_mutex_unlock( &lspmac_moving_mutex);

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);
}

```

5.10.2.10 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 122 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, "
        manualMount", 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_home1_queue( kappa);
    lspmac_home1_queue( omega);
    lspmac_move_or_jog_abs_queue( phi, "
        manualMount", 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, "
        manualMount", 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( aperz,
    "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( aperz,
    "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( aperz,
    "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);
}

```

5.10.2.11 double md2cmds_prep_motion (lspmac_motor_t * mp, double pos)

Definition at line 313 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;
}

```


5.10.2.12 void md2cmds_rotate ()

Spin 360 and make a video (recenter first, maybe)

< velocity (cnts/msec) for omega

Definition at line 601 of file md2cmds.c.

```

    {
int v;
double cx, cy, ax, ay, az;
struct timespec snooze;

//
// 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_mvcenter_prep();
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_mvcenter_wait();
    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;
}

```

5.10.2.13 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 699 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)));
}

```

5.10.2.14 void md2cmds_run()

Start up the thread.

Definition at line 792 of file md2cmds.c.

```

{
    pthread_create( &md2cmds_thread, NULL,
        md2cmds_worker, NULL);
    lsevents_add_listener( "omega crossed zero",
        md2cmds_rotate_cb);
    lsevents_add_listener( "omega In Position",
        md2cmds_maybe_rotate_done_cb);
    lsevents_add_listener( "align.x In Position",
        md2cmds_maybe_done_moving_cb);
    lsevents_add_listener( "align.y In Position",

```

```

    md2cmds_maybe_done_moving_cb);
lsevents_add_listener( "align.z In Position",
    md2cmds_maybe_done_moving_cb);
lsevents_add_listener( "centering.x In Position",
    md2cmds_maybe_done_moving_cb);
lsevents_add_listener( "centering.y In Position",
    md2cmds_maybe_done_moving_cb);
lsevents_add_listener( "align.x Moving",
    md2cmds_maybe_done_moving_cb);
lsevents_add_listener( "align.y Moving",
    md2cmds_maybe_done_moving_cb);
lsevents_add_listener( "align.z Moving",
    md2cmds_maybe_done_moving_cb);
lsevents_add_listener( "centering.x Moving",
    md2cmds_maybe_done_moving_cb);
lsevents_add_listener( "centering.y Moving",
    md2cmds_maybe_done_moving_cb);
lsevents_add_listener( "cam.zoom In Position",
    md2cmds_set_scale_cb);
}

```

5.10.2.15 void md2cmds_set_scale_cb (char * *event*)

Fix up xscale and yscale when zoom changes.

Definition at line 726 of file md2cmds.c.

```

{
    int mag;

    mag = lspmac_getPosition( zoom);
    lspg_query_push( NULL, "SELECT pmac.md2_set_scales( %d)", mag)
    ;
}

```

5.10.2.16 void md2cmds_transfer ()

Transfer a sample TODO: Implement.

Definition at line 29 of file md2cmds.c.

```

{
}

```

5.10.2.17 void* md2cmds_worker (void * *dummy*)

Our worker thread.

Parameters

<i>dummy</i>	[in] Unused but required by protocol
--------------	--------------------------------------

Definition at line 744 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;
}
}

```

5.10.3 Variable Documentation

5.10.3.1 char md2cmds_cmd[MD2CMDS_CMD_LENGTH]

our command;

Definition at line 19 of file md2cmds.c.

5.10.3.2 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.

5.10.3.3 pthread_cond_t md2cmds_moving_cond

coordinate call and response

Definition at line 13 of file md2cmds.c.

5.10.3.4 int md2cmds_moving_count = 0

Definition at line 17 of file md2cmds.c.

5.10.3.5 pthread_mutex_t md2cmds_moving_mutex

message passing between md2cmds and pg

Definition at line 14 of file md2cmds.c.

5.10.3.6 pmac_cmd_queue_t* md2cmds_moving_pq

pmac queue item from last command

Definition at line 15 of file md2cmds.c.

5.10.3.7 pthread_mutex_t md2cmds_mutex

mutex for the condition

Definition at line 11 of file md2cmds.c.

5.10.3.8 pthread_t md2cmds_thread [static]

Definition at line 21 of file md2cmds.c.

5.10.3.9 int rotating = 0 [static]

flag: when omega is in position after a rotate we want to re-home omega

Definition at line 23 of file md2cmds.c.

5.11 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.

5.11.1 Detailed Description

Main for the pgpmac project.

Date

2012

Author

Keith Brister


```

// lsupdate_init();
lskvs_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();

lslogging_run();
lsevents_run();
lsredis_run();
lstimer_run();
lspmac_run();
lspg_run();
// lsupdate_run();
md2cmds_run();

while( 1) {
    //
    // Big loop
    //

    nfd = 0;

    //
    // keyboard
    //
    memcpy( &(fda[nfd++]), &stdinfd, sizeof( struct pollfd));

    if( nfd == 0) {
        //
        // No connectons yet. Wait a bit and try again.
        //
        sleep( 10);
        //
        // go try to connect again
        //
        continue;
    }

    pollrtn = poll( fda, nfd, 10);

    for( i=0; pollrtn>0 && i<nfd; i++) {
        if( fda[i].revents) {
            pollrtn--;
            if( fda[i].fd == 0) {
                stdinService( &fda[i]);
            }
        }
    }
}
}

```

5.11.2.2 void pgpmac_printf (char *fmt, ...)

Terminal output routine ala printf.

Parameters

<i>in</i>	<i>fmt</i>	Printf style formatting string
-----------	------------	--------------------------------

Definition at line 326 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);
    }

```

5.11.2.3 void stdinService (struct pollfd * evt)

Handle keyboard input.

Parameters

<i>in</i>	<i>evt</i>	The pollfd object that caused this call
-----------	------------	---

Definition at line 254 of file pgpmac.c.

```

    {
        static char cmds[1024];
        static char cntrlcmd[2];
        static char cmds_on = 0;
        int ch;

        for( ch=wgetch(term_input); ch != ERR; ch=wgetch(term_input)
        )) {
            // wprintw( term_output, "%04x\n", ch);
            // wnoutrefresh( term_output);

            switch( ch) {
            case KEY_F(1):
                endwin();
                exit(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( 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( 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;
    }

    mvwprintw( term_input, 1, 1, "PMAC> %s", cmds );
    wclrtoeol( term_input );
    box( term_input, 0, 0 );
    wnoutrefresh( term_input );
    doupdate();
}
}

```

5.11.3 Variable Documentation

5.11.3.1 pthread_mutex_t ncurses_mutex

allow more than one thread access to the screen

Definition at line 242 of file pgpmac.c.

5.11.3.2 struct pollfd stdinfd [static]

Handle input from the keyboard.

Definition at line 248 of file pgpmac.c.

5.11.3.3 WINDOW* term_input

place to put the cursor

Definition at line 238 of file pgpmac.c.

5.11.3.4 WINDOW* term_output

place to print stuff out

Definition at line 237 of file pgpmac.c.

5.11.3.5 WINDOW* term_status

shutter, lamp, air, etc status

Definition at line 239 of file pgpmac.c.

5.11.3.6 WINDOW* term_status2

shutter, lamp, air, etc status

Definition at line 240 of file pgpmac.c.

5.12 pgpmac.h File Reference

Headers for the entire pgpmac project.

```
#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>
```

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 [lskvs_kvs_struct](#)
Storage for the key value pairs.
- struct [lskvs_kvs_list_struct](#)
A second linked list type to handle private lists of KVs.
- struct [lspmac_motor_struct](#)
Motor information.
- struct [lspmac_bi_struct](#)
Storage for binary inputs.
- struct [lspg_getcenter_struct](#)
Storage for getcenter query Used for the md2 ROTATE command that generates the centering movies.
- struct [lspg_nextshot_struct](#)
Storage definition for nextshot query.

Macros

- [#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](#) 32
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](#) [lspmac_cmd_queue_t](#)
PMAC command queue item.
- typedef struct [lskvs_kvs_struct](#) [lskvs_kvs_t](#)
Storage for the key value pairs.
- typedef struct [lskvs_kvs_list_struct](#) [lskvs_kvs_list_t](#)
A second linked list type to handle private lists of KVs.
- typedef struct [lspmac_motor_struct](#) [lspmac_motor_t](#)
Motor information.
- typedef struct [lspmac_bi_struct](#) [lspmac_bi_t](#)
Storage for binary inputs.
- 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_nextshot_struct](#) [lspg_nextshot_t](#)
Storage definition for nextshot query.

Functions

- double [lspmac_getPosition](#) ([lspmac_motor_t](#) *)
get the motor position (with locking)
- void [PmacSockSendline](#) (char *s)
- void [pgpmac_printf](#) (char *fmt,...)
Terminal output routine ala printf.
- void [lspg_init](#) ()
Initialize the lspg module.
- 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)
Convenience function to call seq run prep.
- void [lspg_zoom_lut_call](#) ()
- void [lspmac_init](#) (int, int)
Initialize this module.
- void [lspmac_run](#) ()
Start up the lspmac thread.
- void [lspmac_move_or_jog_queue](#) ([lspmac_motor_t](#) *, double, int)

- void [lspmac_move_or_jog_preset_queue](#) ([lspmac_motor_t](#) *, char *, int)
move using a preset value
- void [lspmac_moveabs_queue](#) ([lspmac_motor_t](#) *, double)
Use coordinate system motion program, if available, to move motor to requested position.
- void [lspmac_jogabs_queue](#) ([lspmac_motor_t](#) *, double)
Use jog to move motor to requested position.
- [lpmac_cmd_queue_t](#) * [lspmac_SockSendline](#) (char *,...)
Send a one line command.
- void [lsupdate_init](#) ()
Initialize this module.
- void [md2cmds_init](#) ()
Initialize the md2cmds module.
- void [md2cmds_run](#) ()
Start up the thread.
- void [lsupdate_run](#) ()
run the update routines
- void [lsevents_init](#) ()
Initialize this module.
- void [lsevents_run](#) ()
Start up the thread and get out of the way.
- void [lsevents_send_event](#) (char *,...)
Call the callback routines for the given event.
- void [lsevents_add_listener](#) (char *, void(*cb)(char *))
Add a callback routine to listen for a specific event.
- void [lsevents_remove_listener](#) (char *, void(*cb)(char *))
Remove a listener previously added with lsevents_add_listener.
- void [lstimer_init](#) ()
Initialize the timer list and pthread stuff.
- void [lstimer_run](#) ()
Start up our thread.
- void [lstimer_add_timer](#) (char *, int, unsigned long int, unsigned long int)
Create a timer.
- void [lskvs_regcomp](#) (regex_t *preg, int cflags, char *fmt,...)
Utility wrapper for regcomp providing printf style formatting.
- double [lskvs_find_preset_position](#) ([lspmac_motor_t](#) *mp, char *name, int *err)
find a postion for a given preset name
- void [lsredis_init](#) (char *pub, char *re, char *head)
Initialize this module, that is, set up the connections.
- void [lsredis_run](#) ()
- [lsredis_obj_t](#) * [lsredis_get_obj](#) (char *,...)

Variables

- [lspg_getcenter_t](#) [lspg_getcenter](#)
the getcenter object
- [lspg_nextshot_t](#) [lspg_nextshot](#)
the nextshot object
- [lskvs_kvs_t](#) * [lskvs_kvs](#)
our list (or at least the start of it
- pthread_rwlock_t [lskvs_rwlock](#)

- needed to protect the list*
- [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 * blight_ud](#)
Back light Up/Down actuator.
- [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_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](#) * [fluor](#)
Move the fluorescence detector in/out.
- int [lspmac_nmotors](#)
The number of motors we manage.
- 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;

5.12.1 Detailed Description

Headers for the entire pgpmac project.

Date

2012

Author

Keith Brister

Copyright

All Rights Reserved

Definition in file [pgpmac.h](#).

5.12.2 Macro Definition Documentation

5.12.2.1 `#define LS_DISPLAY_WINDOW_HEIGHT 8`

Number of status box rows.

Definition at line 52 of file pgpmac.h.

5.12.2.2 `#define LS_DISPLAY_WINDOW_WIDTH 24`

Number of status box columns.

Definition at line 56 of file pgpmac.h.

5.12.2.3 `#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 59 of file pgpmac.h.

5.12.2.4 `#define LSEVENTS_EVENT_LENGTH 32`

Fixed length for event names: simplifies string handling.

Definition at line 62 of file pgpmac.h.

5.12.2.5 `#define MD2CMD5_CMD_LENGTH 32`

Definition at line 407 of file pgpmac.h.

5.12.3 Typedef Documentation

5.12.3.1 `typedef struct lskvs_kvs_list_struct lskvs_kvs_list_t`

A second linked list type to handle private lists of KVs.

Developed to support lists of preset motor positions.

5.12.3.2 `typedef struct lskvs_kvs_struct lskvs_kvs_t`

Storage for the key value pairs.

the k's and v's are strings and to keep the memory management less crazy we'll calloc some space for these strings and only free and re-calloc if we need more space later. Only the values are ever going to be resized.

5.12.3.3 `typedef struct lspg_getcenter_struct lspg_getcenter_t`

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

5.12.3.4 `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!)

5.12.3.5 `typedef struct lspmac_bi_struct lspmac_bi_t`

Storage for binary inputs.

5.12.3.6 `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.

5.12.3.7 `typedef struct lsredis_obj_struct lsredis_obj_t`

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

5.12.3.8 `typedef struct lspmac_cmd_queue_struct pmac_cmd_queue_t`

PMAC command queue item.

Command queue items are fixed length to simplify memory management.

5.12.3.9 `typedef struct tagEthernetCmd pmac_cmd_t`

PMAC ethernet packet definition.

Taken directly from the Delta Tau documentation.

5.12.4 Function Documentation

5.12.4.1 `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 77 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);
}

```

5.12.4.2 void lsevents_init ()

Initialize this module.

Definition at line 206 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);
}

```

5.12.4.3 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 122 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);
}
}

```

5.12.4.4 void lsevents_run ()

Start up the thread and get out of the way.

Definition at line 214 of file lsevents.c.

```

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

```

5.12.4.5 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];
char *sp;
va_list arg_ptr;

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

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

pthread_mutex_lock( &lsevents_queue_mutex);

```

```

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

sp = lsevents_queue[(lsevents_queue_on++) %
    LSEVENTS_QUEUE_LENGTH].event;
strncpy( sp, event, LSEVENTS_EVENT_LENGTH);
sp[LSEVENTS_EVENT_LENGTH - 1] = 0;

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

```

5.12.4.6 double lskvs_find_preset_position (lspmac_motor_t * mp, char * name, int * err)

find a postion for a given preset name

Parameters

<i>mp</i>	Motor pointer
<i>name</i>	The preset to search for
<i>err</i>	set to non-zero on error, ignored if null

Definition at line 21 of file lskvs.c.

```

{
    regmatch_t pmatch[4], qmatch[4];
    double rtn;
    lskvs_kvs_list_t
        *position_kv = NULL,
        *name_kv = NULL;
    int e;

    *err = -4;
    if( name == NULL || *name == 0)
        return 0.0;

    *err = 0;
    for( name_kv = mp->presets; name_kv != NULL; name_kv = name_kv->next
    ) {
        if( strcmp( name, name_kv->kvs->v) == 0) {
            //
            // We found the correct preset, now get the index
            //
            e = regexec( &(mp->preset_regex), name_kv->kvs->k, 4, pmatch,
                0);
            if( e != 0) {
                lslogging_log_message( "
lskvs_find_preset_position: could not parse name key '%s'", name_kv->kvs->k);
                if( err != NULL)
                    *err = e;
                return 0.0;
            }

            for( position_kv = mp->presets; position_kv != NULL; position_kv =
                position_kv->next) {
                if( position_kv == name_kv)
                    continue;

                e = regexec( &(mp->preset_regex), position_kv->kvs->k,
                    4, qmatch, 0);
                if( e != 0) {
                    lslogging_log_message( "
lskvs_find_preset_position: could not parse position key '%s'", position_kv->kvs->k);
                    if( err != NULL)
                        *err = e;
                    return 0.0;
                }

                if( strncmp( name_kv->kvs->k, position_kv->kvs->k, qmatch[2].rm_eo
                    + 1) == 0) {
                    break;
                }
            }
            if( position_kv != NULL)

```

```

        break;
    }
}

if( name_kv != NULL || position_kv != NULL) {
    errno = 0;
    rtn = strtod( position_kv->kvs->v, NULL);
    if( errno != 0) {
        lslogging_log_message( "lskvs_find_preset_position:
            bad preset value for motor %s, preset %s, value '%s'", mp->name, name,
            position_kv->kvs->v);
        if( err != NULL)
            *err = -2;
        return 0.0;
    }
    return rtn;
}
lslogging_log_message( "lskvs_find_preset_position:
    could not find preset for motor %s, preset %s", mp->name, name);
if( err != NULL)
    *err = -3;
return 0.0;
}

```

5.12.4.7 void lskvs.regcomp (regex_t * preg, int cflags, char * fmt, ...)

Utility wrapper for regcomp providing printf style formatting.

Parameters

<i>preg</i>	Buffer for the compile regex object
<i>cflags</i>	See regcomp man page
<i>fmt</i>	Printf style formatting string
...	Argument list specified by fmt

< no reason our search strings should ever be this big

Definition at line 92 of file lskvs.c.

```

{
struct regerror_struct {
    int errcode;
    char *errstr;
};
static struct regerror_struct regerrors[] = {
    { REG_BADBR, "Invalid use of back reference operator."},
    { REG_BADPAT, "Invalid use of pattern operators such as group or list."},
    { REG_BADRPT, "Invalid use of repetition operators such as using '*' as
        the first character."},
    { REG_EBRACE, "Un-matched brace interval operators."},
    { REG_EBRACK, "Un-matched bracket list operators."},
    { REG_ECOLLATE, "Invalid collating element."},
    { REG_ECTYPE, "Unknown character class name."},
    { REG_EEND, "Non specific error. This is not defined by POSIX.2."},
    { REG_EESCAPE, "Trailing backslash."},
    { REG_EPAREN, "Un-matched parenthesis group operators."},
    { REG_ERANGE, "Invalid use of the range operator, e.g., the ending point
        of the range occurs prior to the starting point."},
    { REG_ESIZE, "Compiled regular expression requires a pattern buffer
        larger than 64Kb. This is not defined by POSIX.2."},
    { REG_ESPACE, "The regex routines ran out of memory."},
    { REG_ESUBREG, "Invalid back reference to a subexpression."},
    { 0, "No errors"}
};

va_list arg_ptr;
char s[512];
int err;

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

err = regcomp( preg, s, cflags);
if( err != 0) {

```

```

int i;

for( i=0; regerrors[i].errcode != 0; i++)
    if( regerrors[i].errcode == err)
        break;

if( regerrors[i].errcode != 0) {
    lslogging_log_message( "lskvs_regcomp: could not
        compile regular experssion '%s'", s);
    lslogging_log_message( "lskvs_regcomp: regcomp
        returned %d: %s", err, regerrors[i]);
}
}
}

```

5.12.4.8 void lspg_init ()

Initiallize the lspg module.

Definition at line 1666 of file lspg.c.

```

{
    pthread_mutex_init( &lspg_queue_mutex, NULL);
    pthread_cond_init( &lspg_queue_cond, NULL);
    lspg_nextshot_init();
    lspg_getcenter_init();
    lspg_wait_for_detector_init();
    lspg_lock_diffractionmeter_init();
    lspg_lock_detector_init();
}

```

5.12.4.9 void lspg_run ()

Start 'er runnin'.

Definition at line 1678 of file lspg.c.

```

{
    pthread_create( &lspg_thread, NULL, lspg_worker, NULL);
}

```

5.12.4.10 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 986 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();
}

```

```
}

```

5.12.4.11 void lspg_zoom_lut.call ()

5.12.4.12 double lspmac_getPosition (lspmac_motor_t * mp)

get the motor position (with locking)

Parameters

<i>mp</i>	the motor object
-----------	------------------

Definition at line 1230 of file lspmac.c.

```

double rtn;
pthread_mutex_lock( &(mp->mutex));
rtn = mp->position;
pthread_mutex_unlock( &(mp->mutex));
return rtn;
}

```

5.12.4.13 void lspmac_init (int , int)

Initialize this module.

Definition at line 2578 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);

    //
    // Initialize the motor objects
    //

    p = &md2_status;

    omega = lspmac_motor_init( &(lspmac_motors
    [ 0]), 1, 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]), 2, 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]), 3, 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]), 4, 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]), 5, 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]), 6, 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]), 7, 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]), 8, 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]), 9, 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]), 10, 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]), 11, 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]), 17, 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]), 18, 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]), 19, 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]), 20, 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, 160.0, "M1200", "frontLight.intensity",
lspmac_movedac_queue);
blight = lspmac_dac_init( &(lspmac_motors[1
7]), &p->back_dac, 160.0, "M1201", "backLight.intensity",
lspmac_movedac_queue);
fscint = lspmac_dac_init( &(lspmac_motors[1
8]), &p->scint_piezo, 320.0, "M1203", "scint.focus",
lspmac_movedac_queue);

blight_ud = lspmac_bo_init( &(lspmac_motors
[19]), "backLight", "M1101=%d", &(md2_status.accllc_5), 0x02)
;
cryo = lspmac_bo_init( &(lspmac_motors[20
]), "cryo", "M1102=%d", &(md2_status.accllc_5), 0x04);
dryer = lspmac_bo_init( &(lspmac_motors[2
1]), "dryer", "M1103=%d", &(md2_status.accllc_5), 0x08);
fluo = lspmac_bo_init( &(lspmac_motors[22
]), "fluo", "M1008=%d", &(md2_status.accllc_2), 0x01);
flight_oo = lspmac_soft_motor_init( &(
lspmac_motors[23]), "frontLight", 1.0,
lspmac_moveabs_frontlight_oo_queue);
blight_f = lspmac_soft_motor_init( &(
lspmac_motors[24]), "backLight.factor", 1.0,
lspmac_moveabs_blight_factor_queue);
flight_f = lspmac_soft_motor_init( &(
lspmac_motors[25]), "frontLight.factor", 1.0,
lspmac_moveabs_flight_factor_queue);

cryo_switch = lspmac_bi_init( &(lspmac_bis
[0]), &(md2_status.accllc_1), 0x04, "CryoSwitchChanged", "
CryoSwitchChanged");

//
// 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);
}

```

5.12.4.14 void lspmac_jogabs.queue(lspmac_motor_t *, double)

Use jog to move motor to requested position.

Definition at line 2308 of file lspmac.c.

```

{
    lspmac_move_or_jog_abs_queue( mp,
        requested_position, 1);
}

```

5.12.4.15 void lspmac_move_or_jog_preset.queue(lspmac_motor_t *, char *, int)

move using a preset value

Definition at line 2277 of file lspmac.c.

```

{
    double pos;
    int err;

    if( preset == NULL || *preset == 0)
        return;

    pthread_mutex_lock( &(mp->mutex));
    pos = lskvs_find_preset_position( mp, preset, &err);
    ;
    pthread_mutex_unlock( &(mp->mutex));

    lspmac_move_or_jog_abs_queue( mp, pos, use_jog);
}

```

5.12.4.16 void lspmac_move_or_jog.queue(lspmac_motor_t *, double, int)

5.12.4.17 void lspmac_moveabs.queue(lspmac_motor_t *, double)

Use coordinate system motion program, if available, to move motor to requested position.

Definition at line 2298 of file lspmac.c.

```

{
    lspmac_move_or_jog_abs_queue( mp,
        requested_position, 0);
}

```


5.12.4.18 void lspmac_run ()

Start up the lspmac thread.

Definition at line 2792 of file lspmac.c.

```

    {
pthread_create( &pmac_thread, NULL, lspmac_worker,
    NULL);
lsevents_add_listener( "NewKV", lspmac_newKV_cb
    );
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);
    }

```

5.12.4.19 pmac_cmd_queue_t* lspmac_SockSendline(char * fmt, ...)

Send a one line command.

Uses printf style arguments.

Parameters

in	fmt	Printf style format string
----	-----	----------------------------

Definition at line 957 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);
    }

```

5.12.4.20 lsredis_obj_t* lsredis_get_obj(char *, ...)

Definition at line 297 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;
rtn = _lsredis_get_obj( kp);
free( kp);
return rtn;
}

```

5.12.4.21 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 601 of file lsredis.c.

```

{

lsredis_head = strdup( head);
lsredis_publisher = strdup( pub);

pthread_mutex_init( &lsredis_objs_mutex, NULL);
pthread_mutex_init( &lsredis_ro_mutex, NULL);
pthread_mutex_init( &lsredis_wr_mutex, 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);
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;

lsredis_select( re);
}

```

5.12.4.22 void lsredis_run ()

Definition at line 731 of file lsredis.c.

```

    {
        pthread_create( &lsredis_thread, NULL, lsredis_worker
            , NULL);
    }

```

5.12.4.23 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);
    }

```

5.12.4.24 void lstimer_init ()

Initialize the timer list and pthread stuff.

Definition at line 262 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);
    }

```

5.12.4.25 void lstimer_run ()

Start up our thread.

Definition at line 276 of file lstimer.c.

```

    {
        pthread_create( &lstimer_thread, NULL, lstimer_worker
            , NULL);
    }

```

5.12.4.26 void lsupdate_init ()

Initialize this module.

Definition at line 109 of file lsupdate.c.

```

    {

```

5.12.4.27 void lsupdate_run ()

run the update routines

Definition at line 114 of file lsupdate.c.

```

    {
        pthread_create( &lsupdate_thread, NULL, lsupdate_worker
            , NULL);
    }

```

5.12.4.28 void md2cmds_init ()

Initialize the md2cmds module.

Definition at line 778 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);
    }

```

5.12.4.29 void md2cmds_run ()

Start up the thread.

Definition at line 792 of file md2cmds.c.

```

{
pthread_create( &md2cmds_thread, NULL,
md2cmds_worker, NULL);
lsevents_add_listener( "omega crossed zero",
md2cmds_rotate_cb);
lsevents_add_listener( "omega In Position",
md2cmds_maybe_rotate_done_cb);
lsevents_add_listener( "align.x In Position",
md2cmds_maybe_done_moving_cb);
lsevents_add_listener( "align.y In Position",
md2cmds_maybe_done_moving_cb);
lsevents_add_listener( "align.z In Position",
md2cmds_maybe_done_moving_cb);
lsevents_add_listener( "centering.x In Position",
md2cmds_maybe_done_moving_cb);
lsevents_add_listener( "centering.y In Position",
md2cmds_maybe_done_moving_cb);
lsevents_add_listener( "align.x Moving",
md2cmds_maybe_done_moving_cb);
lsevents_add_listener( "align.y Moving",
md2cmds_maybe_done_moving_cb);
lsevents_add_listener( "align.z Moving",
md2cmds_maybe_done_moving_cb);
lsevents_add_listener( "centering.x Moving",
md2cmds_maybe_done_moving_cb);
lsevents_add_listener( "centering.y Moving",
md2cmds_maybe_done_moving_cb);
lsevents_add_listener( "cam.zoom In Position",
md2cmds_set_scale_cb);
}

```

5.12.4.30 void pgpmac_printf (char *fmt, ...)

Terminal output routine ala printf.

Parameters

in	fmt	Printf style forming string
----	-----	-----------------------------

Definition at line 326 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);
}

```

5.12.4.31 void PmacSockSendline (char *s)

5.12.5 Variable Documentation

5.12.5.1 lspmac_motor_t* alignx

Alignment stage X.

Definition at line 81 of file lspmac.c.

5.12.5.2 **lspmac_motor_t*** aligny

Alignment stage Y.

Definition at line 82 of file lspmac.c.

5.12.5.3 **lspmac_motor_t*** alignz

Alignment stage X.

Definition at line 83 of file lspmac.c.

5.12.5.4 **lspmac_motor_t*** anal

Polaroid analyzer motor.

Definition at line 84 of file lspmac.c.

5.12.5.5 **lspmac_motor_t*** apery

Aperture Y.

Definition at line 86 of file lspmac.c.

5.12.5.6 **lspmac_motor_t*** aperz

Aperture Z.

Definition at line 87 of file lspmac.c.

5.12.5.7 **lspmac_motor_t*** blight

Back Light DAC.

Definition at line 98 of file lspmac.c.

5.12.5.8 **lspmac_motor_t*** blight_f

Back light scale factor.

Definition at line 103 of file lspmac.c.

5.12.5.9 **lspmac_motor_t*** blight_ud

Back light Up/Down actuator.

Definition at line 101 of file lspmac.c.

5.12.5.10 **lspmac_motor_t*** capy

Capillary Y.

Definition at line 88 of file lspmac.c.

5.12.5.11 lspmac_motor_t* capz

Capillary Z.

Definition at line 89 of file lspmac.c.

5.12.5.12 lspmac_motor_t* cenx

Centering Table X.

Definition at line 91 of file lspmac.c.

5.12.5.13 lspmac_motor_t* ceny

Centering Table Y.

Definition at line 92 of file lspmac.c.

5.12.5.14 lspmac_motor_t* cryo

Move the cryostream towards or away from the crystal.

Definition at line 105 of file lspmac.c.

5.12.5.15 lspmac_motor_t* dryer

blow air on the scintillator to dry it off

Definition at line 106 of file lspmac.c.

5.12.5.16 lspmac_motor_t* flight

Front Light DAC.

Definition at line 97 of file lspmac.c.

5.12.5.17 lspmac_motor_t* flight_f

Front light scale factor.

Definition at line 104 of file lspmac.c.

5.12.5.18 lspmac_motor_t* flight_oo

Turn front light on/off.

Definition at line 102 of file lspmac.c.

5.12.5.19 lspmac_motor_t* fluo

Move the fluorescence detector in/out.

Definition at line 107 of file lspmac.c.

5.12.5.20 `lspmac_motor_t* fscint`

Scintillator Piezo DAC.

Definition at line 99 of file `lspmac.c`.

5.12.5.21 `lspmac_motor_t* fshut`

Fast shutter.

Definition at line 96 of file `lspmac.c`.

5.12.5.22 `lspmac_motor_t* kappa`

Kappa.

Definition at line 93 of file `lspmac.c`.

5.12.5.23 `lskvs_kvs_t* lskvs_kvs`

our list (or at least the start of it

Definition at line 11 of file `lskvs.c`.

5.12.5.24 `pthread_rwlock_t lskvs_rwlock`

needed to protect the list

Definition at line 12 of file `lskvs.c`.

5.12.5.25 `lspg_getcenter_t lspg_getcenter`

the getcenter object

Definition at line 73 of file `lspg.c`.

5.12.5.26 `lspg_nextshot_t lspg_nextshot`

the nextshot object

Definition at line 72 of file `lspg.c`.

5.12.5.27 `lspmac_motor_t lspmac_motors[]`

All our motors.

Definition at line 78 of file `lspmac.c`.

5.12.5.28 `pthread_cond_t lspmac_moving_cond`

Wait for motor(s) to finish moving condition.

Definition at line 58 of file `lspmac.c`.

5.12.5.29 int lspmac_moving_flags

Flag used to implement motor moving condition.

Definition at line 59 of file lspmac.c.

5.12.5.30 pthread_mutex_t lspmac_moving_mutex

Coordinate moving motors between threads.

Definition at line 57 of file lspmac.c.

5.12.5.31 int lspmac_nmotors

The number of motors we manage.

Definition at line 79 of file lspmac.c.

5.12.5.32 pthread_cond_t lspmac_shutter_cond

Allows waiting for the shutter status to change.

Definition at line 56 of file lspmac.c.

5.12.5.33 int lspmac_shutter_has_opened

Indicates that the shutter had opened, perhaps briefly even if the state did not change.

Definition at line 54 of file lspmac.c.

5.12.5.34 pthread_mutex_t lspmac_shutter_mutex

Coordinates threads reading shutter status.

Definition at line 55 of file lspmac.c.

5.12.5.35 int lspmac_shutter_state

State of the shutter, used to detect changes.

Definition at line 53 of file lspmac.c.

5.12.5.36 pthread_mutex_t md2_status_mutex

Synchronize reading/writing status buffer.

Definition at line 296 of file lspmac.c.

5.12.5.37 char md2cmds_cmd[]

our command;

Definition at line 19 of file md2cmds.c.

5.12.5.38 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.

5.12.5.39 pthread_mutex_t md2cmds_mutex

mutex for the condition

Definition at line 11 of file md2cmds.c.

5.12.5.40 pthread_cond_t md2cmds_pg_cond**5.12.5.41 pthread_mutex_t md2cmds_pg_mutex****5.12.5.42 pthread_mutex_t ncurses_mutex**

allow more than one thread access to the screen

Definition at line 242 of file pgpmac.c.

5.12.5.43 lspmac_motor_t* omega

MD2 omega axis (the air bearing)

Definition at line 80 of file lspmac.c.

5.12.5.44 struct timespec omega_zero_time

Time we believe that omega crossed zero.

Definition at line 63 of file lspmac.c.

5.12.5.45 lspmac_motor_t* phi

Phi (not data collection axis)

Definition at line 94 of file lspmac.c.

5.12.5.46 pthread_cond_t pmac_queue_cond

wait for a command to be sent to PMAC before continuing

Definition at line 69 of file lspmac.c.

5.12.5.47 pthread_mutex_t pmac_queue_mutex

manage access to the pmac command queue

Definition at line 68 of file lspmac.c.

5.12.5.48 lspmac_motor_t* scint

Scintillator Z.

Definition at line 90 of file lspmac.c.

5.12.5.49 WINDOW* term_input

place to put the cursor

Definition at line 238 of file pgpmac.c.

5.12.5.50 WINDOW* term_output

place to print stuff out

Definition at line 237 of file pgpmac.c.

5.12.5.51 WINDOW* term_status

shutter, lamp, air, etc status

Definition at line 239 of file pgpmac.c.

5.12.5.52 WINDOW* term_status2

shutter, lamp, air, etc status

Definition at line 240 of file pgpmac.c.

5.12.5.53 lspmac_motor_t* zoom

Optical zoom.

Definition at line 85 of file lspmac.c.

Index

- [_lsredis_get_obj](#)
 - [lsredis.c, 180](#)
 - [_lsredis_set_value](#)
 - [lsredis.c, 181](#)
- [acc11c_1](#)
 - [md2StatusStruct, 45](#)
- [acc11c_2](#)
 - [md2StatusStruct, 45](#)
- [acc11c_3](#)
 - [md2StatusStruct, 45](#)
- [acc11c_5](#)
 - [md2StatusStruct, 45](#)
- [acc11c_6](#)
 - [md2StatusStruct, 45](#)
- [active](#)
 - [lspg_nextshot_struct, 20](#)
- [active2](#)
 - [lspg_nextshot_struct, 20](#)
- [active2_isnull](#)
 - [lspg_nextshot_struct, 20](#)
- [active_isnull](#)
 - [lspg_nextshot_struct, 20](#)
- [actual_pos_cnts](#)
 - [lspmac_motor_struct, 34](#)
- [actual_pos_cnts_p](#)
 - [lspmac_motor_struct, 34](#)
- [addRead](#)
 - [kvredis.c, 56](#)
- [addWrite](#)
 - [kvredis.c, 56](#)
- [alignx](#)
 - [lspmac.c, 171](#)
 - [pgpmac.h, 241](#)
- [alignx_act_pos](#)
 - [md2StatusStruct, 45](#)
- [alignx_status_1](#)
 - [md2StatusStruct, 45](#)
- [alignx_status_2](#)
 - [md2StatusStruct, 45](#)
- [aligny](#)
 - [lspmac.c, 171](#)
 - [pgpmac.h, 242](#)
- [aligny_act_pos](#)
 - [md2StatusStruct, 45](#)
- [aligny_status_1](#)
 - [md2StatusStruct, 45](#)
- [aligny_status_2](#)
 - [md2StatusStruct, 45](#)
- [alignz](#)
 - [lspmac.c, 171](#)
 - [pgpmac.h, 242](#)
- [alignz_act_pos](#)
 - [md2StatusStruct, 45](#)
- [alignz_status_1](#)
 - [md2StatusStruct, 45](#)
- [alignz_status_2](#)
 - [md2StatusStruct, 46](#)
- [anal](#)
 - [lspmac.c, 172](#)
 - [pgpmac.h, 242](#)
- [analyzer_act_pos](#)
 - [md2StatusStruct, 46](#)
- [analyzer_status_1](#)
 - [md2StatusStruct, 46](#)
- [analyzer_status_2](#)
 - [md2StatusStruct, 46](#)
- [aperturey_act_pos](#)
 - [md2StatusStruct, 46](#)
- [aperturey_status_1](#)
 - [md2StatusStruct, 46](#)
- [aperturey_status_2](#)
 - [md2StatusStruct, 46](#)
- [aperturez_act_pos](#)
 - [md2StatusStruct, 46](#)
- [aperturez_status_1](#)
 - [md2StatusStruct, 46](#)
- [aperturez_status_2](#)
 - [md2StatusStruct, 46](#)
- [apery](#)
 - [lspmac.c, 172](#)
 - [pgpmac.h, 242](#)
- [aperz](#)
 - [lspmac.c, 172](#)
 - [pgpmac.h, 242](#)
- [ax](#)
 - [lspg_nextshot_struct, 20](#)
- [ax2](#)
 - [lspg_nextshot_struct, 20](#)
- [ax2_isnull](#)
 - [lspg_nextshot_struct, 20](#)
- [ax_isnull](#)
 - [lspg_nextshot_struct, 20](#)
- [axis](#)
 - [lspmac_motor_struct, 34](#)
- [ay](#)
 - [lspg_nextshot_struct, 20](#)
- [ay2](#)
 - [lspg_nextshot_struct, 20](#)

ay2_isnull
 lspg_nextshot_struct, 20
ay_isnull
 lspg_nextshot_struct, 21
az
 lspg_nextshot_struct, 21
az2
 lspg_nextshot_struct, 21
az2_isnull
 lspg_nextshot_struct, 21
az_isnull
 lspg_nextshot_struct, 21

bData
 tagEthernetCmd, 51
back_dac
 md2StatusStruct, 46
blight
 lspmac.c, 172
 pgpmac.h, 242
blight_f
 lspmac.c, 172
 pgpmac.h, 242
blight_ud
 lspmac.c, 172
 pgpmac.h, 242

capy
 lspmac.c, 172
 pgpmac.h, 242
capy_act_pos
 md2StatusStruct, 46
capy_status_1
 md2StatusStruct, 47
capy_status_2
 md2StatusStruct, 47
capz
 lspmac.c, 172
 pgpmac.h, 242
capz_act_pos
 md2StatusStruct, 47
capz_status_1
 md2StatusStruct, 47
capz_status_2
 md2StatusStruct, 47
cb
 lsevents_listener_struct, 9
centerx_act_pos
 md2StatusStruct, 47
centerx_status_1
 md2StatusStruct, 47
centerx_status_2
 md2StatusStruct, 47
centery_act_pos
 md2StatusStruct, 47
centery_status_1
 md2StatusStruct, 47
centery_status_2
 md2StatusStruct, 47

cenx
 lspmac.c, 172
 pgpmac.h, 243
ceny
 lspmac.c, 172
 pgpmac.h, 243
changeEventOff
 lspmac_bi_struct, 30
changeEventOn
 lspmac_bi_struct, 30
cleanstr
 lspmac.c, 130
cleanup
 kvredis.c, 56
cmdac
 kvredis.c, 67
cmdfd
 kvredis.c, 67
cond
 lspg_getcenter_struct, 14
 lspg_lock_detector_struct, 16
 lspg_lock_diffractionmeter_struct, 16
 lspg_nextshot_struct, 21
 lspg_seq_run_prep_struct, 28
 lspg_wait_for_detector_struct, 29
 lspmac_motor_struct, 34
 lsredis_obj_struct, 40
coord_num
 lspmac_motor_struct, 34
cr_cmd
 lspmac.c, 173
cryo
 lspmac.c, 173
 pgpmac.h, 243
cryo_switch
 lspmac.c, 173
cx
 lspg_nextshot_struct, 21
cx2
 lspg_nextshot_struct, 21
cx2_isnull
 lspg_nextshot_struct, 21
cx_isnull
 lspg_nextshot_struct, 21
cy
 lspg_nextshot_struct, 21
cy2
 lspg_nextshot_struct, 22
cy2_isnull
 lspg_nextshot_struct, 22
cy_isnull
 lspg_nextshot_struct, 22

dac_mvar
 lspmac_motor_struct, 35
dax
 lspg_getcenter_struct, 14
dax_isnull
 lspg_getcenter_struct, 14

- day
 - lspg_getcenter_struct, [14](#)
- day_isnull
 - lspg_getcenter_struct, [14](#)
- daz
 - lspg_getcenter_struct, [14](#)
- daz_isnull
 - lspg_getcenter_struct, [14](#)
- dbmem
 - lspmac.c, [173](#)
- dbmemIn
 - lspmac.c, [173](#)
- dcx
 - lspg_getcenter_struct, [14](#)
- dcx_isnull
 - lspg_getcenter_struct, [15](#)
- dcy
 - lspg_getcenter_struct, [15](#)
- dcy_isnull
 - lspg_getcenter_struct, [15](#)
- debugCB
 - kvredis.c, [56](#)
- delRead
 - kvredis.c, [57](#)
- delWrite
 - kvredis.c, [57](#)
- delay_nsecs
 - lstimer_list_struct, [42](#)
- delay_secs
 - lstimer_list_struct, [42](#)
- dryer
 - lspmac.c, [173](#)
 - pgpmac.h, [243](#)
- dsdir
 - lspg_nextshot_struct, [22](#)
- dsdir_isnull
 - lspg_nextshot_struct, [22](#)
- dsdist
 - lspg_nextshot_struct, [22](#)
- dsdist2
 - lspg_nextshot_struct, [22](#)
- dsdist2_isnull
 - lspg_nextshot_struct, [22](#)
- dsdist_isnull
 - lspg_nextshot_struct, [22](#)
- dsexp
 - lspg_nextshot_struct, [22](#)
- dsexp2
 - lspg_nextshot_struct, [23](#)
- dsexp2_isnull
 - lspg_nextshot_struct, [23](#)
- dsexp_isnull
 - lspg_nextshot_struct, [23](#)
- dshpid
 - lspg_nextshot_struct, [23](#)
- dshpid_isnull
 - lspg_nextshot_struct, [23](#)
- dskappa
 - lspg_nextshot_struct, [23](#)
- dskappa2
 - lspg_nextshot_struct, [23](#)
- dskappa2_isnull
 - lspg_nextshot_struct, [23](#)
- dskappa_isnull
 - lspg_nextshot_struct, [23](#)
- dsnrg
 - lspg_nextshot_struct, [23](#)
- dsnrg2
 - lspg_nextshot_struct, [24](#)
- dsnrg2_isnull
 - lspg_nextshot_struct, [24](#)
- dsnrg_isnull
 - lspg_nextshot_struct, [24](#)
- dsomega
 - lspg_nextshot_struct, [24](#)
- dsomega2
 - lspg_nextshot_struct, [24](#)
- dsomega2_isnull
 - lspg_nextshot_struct, [24](#)
- dsomega_isnull
 - lspg_nextshot_struct, [24](#)
- dsoscaxis
 - lspg_nextshot_struct, [24](#)
- dsoscaxis2
 - lspg_nextshot_struct, [24](#)
- dsoscaxis2_isnull
 - lspg_nextshot_struct, [24](#)
- dsoscaxis_isnull
 - lspg_nextshot_struct, [25](#)
- dsowidth
 - lspg_nextshot_struct, [25](#)
- dsowidth2
 - lspg_nextshot_struct, [25](#)
- dsowidth2_isnull
 - lspg_nextshot_struct, [25](#)
- dsowidth_isnull
 - lspg_nextshot_struct, [25](#)
- dsphi
 - lspg_nextshot_struct, [25](#)
- dsphi2
 - lspg_nextshot_struct, [25](#)
- dsphi2_isnull
 - lspg_nextshot_struct, [25](#)
- dsphi_isnull
 - lspg_nextshot_struct, [25](#)
- dspid
 - lspg_nextshot_struct, [25](#)
- dspid_isnull
 - lspg_nextshot_struct, [25](#)
- dummy1
 - md2StatusStruct, [47](#)
- dummy2
 - md2StatusStruct, [48](#)
- dummy3
 - md2StatusStruct, [48](#)
- dummy4

- md2StatusStruct, 48
- dummy5
 - md2StatusStruct, 48
- dummy6
 - md2StatusStruct, 48
- dummy7
 - md2StatusStruct, 48
- dummy8
 - md2StatusStruct, 48
- dummy9
 - md2StatusStruct, 48
- dummyA
 - md2StatusStruct, 48
- dummyB
 - md2StatusStruct, 48
- ethCmdOff
 - lspmac.c, 173
- ethCmdOn
 - lspmac.c, 173
- ethCmdQueue
 - lspmac.c, 173
- ethCmdReply
 - lspmac.c, 174
- event
 - lsevents_queue_struct, 10
 - lstimer_list_struct, 42
- events_name
 - lsredis_obj_struct, 40
- fd_service
 - kvredis.c, 57
- first_time
 - lspmac_bi_struct, 30
- flight
 - lspmac.c, 174
 - pgpmac.h, 243
- flight_f
 - lspmac.c, 174
 - pgpmac.h, 243
- flight_oo
 - lspmac.c, 174
 - pgpmac.h, 243
- fluo
 - lspmac.c, 174
 - pgpmac.h, 243
- format
 - lspmac_motor_struct, 35
- front_dac
 - md2StatusStruct, 48
- fs_has_opened
 - md2StatusStruct, 48
- fs_has_opened_globally
 - md2StatusStruct, 49
- fs_is_open
 - md2StatusStruct, 49
- fscint
 - lspmac.c, 174
 - pgpmac.h, 243
- fshut
 - lspmac.c, 174
 - pgpmac.h, 244
- gb_cmd
 - lspmac.c, 174
- getd
 - lsredis_obj_struct, 40
- getivars
 - lspmac.c, 174
- getl
 - lsredis_obj_struct, 40
- getmvars
 - lspmac.c, 175
- getstr
 - lsredis_obj_struct, 40
- handler
 - lstimer.c, 195
- hex_dump
 - lspmac.c, 130
- hits
 - lsredis_obj_struct, 40
- home
 - lspmac_motor_struct, 35
- homing
 - lspmac_motor_struct, 35
- init_nsecs
 - lstimer_list_struct, 42
- init_secs
 - lstimer_list_struct, 42
- k
 - lskvs_kvs_struct, 12
- kappa
 - lspmac.c, 175
 - pgpmac.h, 244
- kappa_act_pos
 - md2StatusStruct, 49
- kappa_status_1
 - md2StatusStruct, 49
- kappa_status_2
 - md2StatusStruct, 49
- key
 - lsredis_obj_struct, 40
- kvredis.c, 53
 - addRead, 56
 - addWrite, 56
 - cleanup, 56
 - cmdac, 67
 - cmdfd, 67
 - debugCB, 56
 - delRead, 57
 - delWrite, 57
 - fd_service, 57
 - kvseq, 67
 - LS_PG_STATE_IDLE, 55
 - LS_PG_STATE_INIT, 55

- LS_PG_STATE_RECV, [55](#)
- LS_PG_STATE_RESET, [55](#)
- LS_PG_STATE_SEND, [55](#)
- ls_pg_state, [67](#)
- lspg_allkvs_cb, [58](#)
- lspg_connectPoll_response, [67](#)
- lspg_flush, [58](#)
- lspg_next_state, [59](#)
- lspg_notice_processor, [59](#)
- lspg_pg_connect, [60](#)
- lspg_pg_service, [61](#)
- lspg_query_next, [62](#)
- lspg_query_push, [62](#)
- lspg_query_queue, [67](#)
- lspg_query_queue_off, [67](#)
- lspg_query_queue_on, [68](#)
- lspg_query_queue_reply, [68](#)
- lspg_query_queue_t, [56](#)
- lspg_query_reply_next, [63](#)
- lspg_query_reply_peek, [63](#)
- lspg_receive, [64](#)
- lspg_resetPoll_response, [68](#)
- lspg_send_next_query, [64](#)
- lspgfd, [68](#)
- main, [65](#)
- now, [68](#)
- q, [68](#)
- redisDisconnectCB, [67](#)
- subac, [68](#)
- subfd, [68](#)
- kvs
 - lskvs_kvs_list_struct, [11](#)
- kvseq
 - kvredis.c, [67](#)
- l
 - lskvs_kvs_struct, [12](#)
- LS_PG_STATE_IDLE
 - kvredis.c, [55](#)
 - lspg.c, [89](#)
- LS_PG_STATE_INIT
 - kvredis.c, [55](#)
 - lspg.c, [89](#)
- LS_PG_STATE_RECV
 - kvredis.c, [55](#)
 - lspg.c, [89](#)
- LS_PG_STATE_RESET
 - kvredis.c, [55](#)
 - lspg.c, [89](#)
- LS_PG_STATE_SEND
 - kvredis.c, [55](#)
 - lspg.c, [89](#)
- LS_PMAC_STATE_CR
 - lspmac.c, [126](#)
- LS_PMAC_STATE_GB
 - lspmac.c, [126](#)
- LS_PMAC_STATE_GMR
 - lspmac.c, [126](#)
- LS_PMAC_STATE_IDLE
 - lspmac.c, [127](#)
- LS_PMAC_STATE_RESET
 - lspmac.c, [127](#)
- LS_PMAC_STATE_RR
 - lspmac.c, [127](#)
- LS_PMAC_STATE_SC
 - lspmac.c, [127](#)
- LS_PMAC_STATE_WACK
 - lspmac.c, [127](#)
- LS_PMAC_STATE_WCR
 - lspmac.c, [127](#)
- LS_PMAC_STATE_WGB
 - lspmac.c, [127](#)
- LSLOGGING_FILE_NAME
 - lslogging.c, [81](#)
- LSPMAC_PRESET_REGEX
 - lspmac.c, [127](#)
- LSTIMER_LIST_LENGTH
 - lstimer.c, [194](#)
- last_nsecs
 - lstimer_list_struct, [42](#)
- last_secs
 - lstimer_list_struct, [42](#)
- linesReceived
 - lspmac.c, [175](#)
- lmsg
 - lslogging_queue_struct, [13](#)
- ls_pg_state
 - kvredis.c, [67](#)
 - lspg.c, [117](#)
- ls_pmac_state
 - lspmac.c, [175](#)
- lsConnect
 - lspmac.c, [130](#)
- lsevents.c, [69](#)
 - lsevents_add_listener, [70](#)
 - lsevents_init, [71](#)
 - lsevents_listener_mutex, [73](#)
 - lsevents_listener_t, [70](#)
 - lsevents_listeners_p, [74](#)
 - lsevents_queue, [74](#)
 - lsevents_queue_cond, [74](#)
 - lsevents_queue_mutex, [74](#)
 - lsevents_queue_off, [74](#)
 - lsevents_queue_on, [74](#)
 - lsevents_queue_t, [70](#)
 - lsevents_remove_listener, [71](#)
 - lsevents_run, [72](#)
 - lsevents_send_event, [72](#)
 - lsevents_thread, [74](#)
 - lsevents_worker, [73](#)
- lsevents_add_listener
 - lsevents.c, [70](#)
 - pgpmac.h, [228](#)
- lsevents_init
 - lsevents.c, [71](#)
 - pgpmac.h, [229](#)
- lsevents_listener_mutex

- lsevents.c, [73](#)
- lsevents_listener_struct, [9](#)
 - cb, [9](#)
 - next, [9](#)
 - raw_regexp, [9](#)
 - re, [10](#)
- lsevents_listener_t
 - lsevents.c, [70](#)
- lsevents_listeners_p
 - lsevents.c, [74](#)
- lsevents_queue
 - lsevents.c, [74](#)
- lsevents_queue_cond
 - lsevents.c, [74](#)
- lsevents_queue_mutex
 - lsevents.c, [74](#)
- lsevents_queue_off
 - lsevents.c, [74](#)
- lsevents_queue_on
 - lsevents.c, [74](#)
- lsevents_queue_struct, [10](#)
 - event, [10](#)
- lsevents_queue_t
 - lsevents.c, [70](#)
- lsevents_remove_listener
 - lsevents.c, [71](#)
 - pgpmac.h, [229](#)
- lsevents_run
 - lsevents.c, [72](#)
 - pgpmac.h, [230](#)
- lsevents_send_event
 - lsevents.c, [72](#)
 - pgpmac.h, [230](#)
- lsevents_thread
 - lsevents.c, [74](#)
- lsevents_worker
 - lsevents.c, [73](#)
- lskvs.c, [74](#)
 - lskvs_find_preset_position, [75](#)
 - lskvs_get, [76](#)
 - lskvs_init, [77](#)
 - lskvs_kvs, [80](#)
 - lskvs_regcomp, [77](#)
 - lskvs_run, [78](#)
 - lskvs_rwlock, [80](#)
 - lskvs_set, [78](#)
- lskvs_find_preset_position
 - lskvs.c, [75](#)
 - pgpmac.h, [231](#)
- lskvs_get
 - lskvs.c, [76](#)
- lskvs_init
 - lskvs.c, [77](#)
- lskvs_kvs
 - lskvs.c, [80](#)
 - pgpmac.h, [244](#)
- lskvs_kvs_list_struct, [10](#)
 - kvs, [11](#)
 - next, [11](#)
- lskvs_kvs_list_t
 - pgpmac.h, [227](#)
- lskvs_kvs_struct, [11](#)
 - k, [12](#)
 - l, [12](#)
 - next, [12](#)
 - v, [12](#)
 - vl, [12](#)
- lskvs_kvs_t
 - pgpmac.h, [227](#)
- lskvs_regcomp
 - lskvs.c, [77](#)
 - pgpmac.h, [232](#)
- lskvs_run
 - lskvs.c, [78](#)
- lskvs_rwlock
 - lskvs.c, [80](#)
 - pgpmac.h, [244](#)
- lskvs_set
 - lskvs.c, [78](#)
- lslogging.c, [80](#)
 - LSLOGGING_FILE_NAME, [81](#)
 - lslogging_cond, [83](#)
 - lslogging_file, [83](#)
 - lslogging_init, [82](#)
 - lslogging_log_message, [82](#)
 - lslogging_mutex, [83](#)
 - lslogging_off, [83](#)
 - lslogging_on, [84](#)
 - lslogging_queue, [84](#)
 - lslogging_queue_t, [82](#)
 - lslogging_run, [82](#)
 - lslogging_thread, [84](#)
 - lslogging_worker, [83](#)
- lslogging_cond
 - lslogging.c, [83](#)
- lslogging_file
 - lslogging.c, [83](#)
- lslogging_init
 - lslogging.c, [82](#)
- lslogging_log_message
 - lslogging.c, [82](#)
- lslogging_mutex
 - lslogging.c, [83](#)
- lslogging_off
 - lslogging.c, [83](#)
- lslogging_on
 - lslogging.c, [84](#)
- lslogging_queue
 - lslogging.c, [84](#)
- lslogging_queue_struct, [12](#)
 - lmsg, [13](#)
 - ltime, [13](#)
- lslogging_queue_t
 - lslogging.c, [82](#)
- lslogging_run
 - lslogging.c, [82](#)

- lslogging_thread
 - lslogging.c, 84
- lslogging_worker
 - lslogging.c, 83
- lspg.c, 84
 - LS_PG_STATE_IDLE, 89
 - LS_PG_STATE_INIT, 89
 - LS_PG_STATE_RECV, 89
 - LS_PG_STATE_RESET, 89
 - LS_PG_STATE_SEND, 89
 - ls_pg_state, 117
 - lspg_array2ptrs, 90
 - lspg_blight_lut_cb, 91
 - lspg_cmd_cb, 92
 - lspg_connectPoll_response, 117
 - lspg_flight_lut_cb, 92
 - lspg_flush, 93
 - lspg_getcenter, 117
 - lspg_getcenter_all, 93
 - lspg_getcenter_call, 93
 - lspg_getcenter_cb, 94
 - lspg_getcenter_done, 94
 - lspg_getcenter_init, 95
 - lspg_getcenter_wait, 95
 - lspg_init, 95
 - lspg_init_motors_cb, 95
 - lspg_kvs_cb, 96
 - lspg_lock_detector, 117
 - lspg_lock_detector_all, 97
 - lspg_lock_detector_call, 97
 - lspg_lock_detector_cb, 97
 - lspg_lock_detector_done, 97
 - lspg_lock_detector_init, 98
 - lspg_lock_detector_t, 89
 - lspg_lock_detector_wait, 98
 - lspg_lock_diffractionmeter, 117
 - lspg_lock_diffractionmeter_all, 98
 - lspg_lock_diffractionmeter_call, 98
 - lspg_lock_diffractionmeter_cb, 98
 - lspg_lock_diffractionmeter_done, 99
 - lspg_lock_diffractionmeter_init, 99
 - lspg_lock_diffractionmeter_t, 89
 - lspg_lock_diffractionmeter_wait, 99
 - lspg_next_state, 99
 - lspg_nextaction_cb, 100
 - lspg_nextshot, 117
 - lspg_nextshot_call, 101
 - lspg_nextshot_cb, 101
 - lspg_nextshot_done, 105
 - lspg_nextshot_init, 105
 - lspg_nextshot_wait, 105
 - lspg_notice_processor, 105
 - lspg_pg_connect, 105
 - lspg_pg_service, 107
 - lspg_query_next, 108
 - lspg_query_push, 109
 - lspg_query_queue, 118
 - lspg_query_queue_off, 118
 - lspg_query_queue_on, 118
 - lspg_query_queue_reply, 118
 - lspg_query_queue_t, 89
 - lspg_query_reply_next, 109
 - lspg_query_reply_peek, 109
 - lspg_queue_cond, 118
 - lspg_queue_mutex, 118
 - lspg_receive, 110
 - lspg_resetPoll_response, 118
 - lspg_run, 111
 - lspg_scint_lut_cb, 111
 - lspg_send_next_query, 111
 - lspg_seq_run_prep, 118
 - lspg_seq_run_prep_all, 112
 - lspg_seq_run_prep_call, 112
 - lspg_seq_run_prep_cb, 113
 - lspg_seq_run_prep_done, 113
 - lspg_seq_run_prep_init, 113
 - lspg_seq_run_prep_t, 90
 - lspg_seq_run_prep_wait, 113
 - lspg_sig_service, 114
 - lspg_thread, 118
 - lspg_wait_for_detector, 119
 - lspg_wait_for_detector_all, 114
 - lspg_wait_for_detector_call, 114
 - lspg_wait_for_detector_cb, 114
 - lspg_wait_for_detector_done, 115
 - lspg_wait_for_detector_init, 115
 - lspg_wait_for_detector_t, 90
 - lspg_wait_for_detector_wait, 115
 - lspg_worker, 115
 - lspg_zoom_lut_cb, 116
- lspgfd, 119
- now, 119
- q, 119
- lspg_allkvs_cb
 - kvredis.c, 58
- lspg_array2ptrs
 - lspg.c, 90
- lspg_blight_lut_cb
 - lspg.c, 91
- lspg_cmd_cb
 - lspg.c, 92
- lspg_connectPoll_response
 - kvredis.c, 67
 - lspg.c, 117
- lspg_flight_lut_cb
 - lspg.c, 92
- lspg_flush
 - kvredis.c, 58
 - lspg.c, 93
- lspg_getcenter
 - lspg.c, 117
 - pgpmac.h, 244
- lspg_getcenter_all
 - lspg.c, 93
- lspg_getcenter_call
 - lspg.c, 93

lspg_getcenter_cb
lspg.c, 94

lspg_getcenter_done
lspg.c, 94

lspg_getcenter_init
lspg.c, 95

lspg_getcenter_struct, 13
cond, 14
dax, 14
dax_isnull, 14
day, 14
day_isnull, 14
daz, 14
daz_isnull, 14
dcx, 14
dcx_isnull, 15
dcy, 15
dcy_isnull, 15
mutex, 15
new_value_ready, 15
no_rows_returned, 15
zoom, 15
zoom_isnull, 15

lspg_getcenter_t
pgpmac.h, 228

lspg_getcenter_wait
lspg.c, 95

lspg_init
lspg.c, 95
pgpmac.h, 233

lspg_init_motors_cb
lspg.c, 95

lspg_initialized
lspmac_motor_struct, 35

lspg_kvs_cb
lspg.c, 96

lspg_lock_detector
lspg.c, 117

lspg_lock_detector_all
lspg.c, 97

lspg_lock_detector_call
lspg.c, 97

lspg_lock_detector_cb
lspg.c, 97

lspg_lock_detector_done
lspg.c, 97

lspg_lock_detector_init
lspg.c, 98

lspg_lock_detector_struct, 15
cond, 16
mutex, 16
new_value_ready, 16

lspg_lock_detector_t
lspg.c, 89

lspg_lock_detector_wait
lspg.c, 98

lspg_lock_diffractionmeter
lspg.c, 117

lspg_lock_diffractionmeter_all
lspg.c, 98

lspg_lock_diffractionmeter_call
lspg.c, 98

lspg_lock_diffractionmeter_cb
lspg.c, 98

lspg_lock_diffractionmeter_done
lspg.c, 99

lspg_lock_diffractionmeter_init
lspg.c, 99

lspg_lock_diffractionmeter_struct, 16
cond, 16
mutex, 16
new_value_ready, 17

lspg_lock_diffractionmeter_t
lspg.c, 89

lspg_lock_diffractionmeter_wait
lspg.c, 99

lspg_next_state
kvrodis.c, 59
lspg.c, 99

lspg_nextaction_cb
lspg.c, 100

lspg_nextshot
lspg.c, 117
pgpmac.h, 244

lspg_nextshot_call
lspg.c, 101

lspg_nextshot_cb
lspg.c, 101

lspg_nextshot_done
lspg.c, 105

lspg_nextshot_init
lspg.c, 105

lspg_nextshot_struct, 17
active, 20
active2, 20
active2_isnull, 20
active_isnull, 20
ax, 20
ax2, 20
ax2_isnull, 20
ax_isnull, 20
ay, 20
ay2, 20
ay2_isnull, 20
ay_isnull, 21
az, 21
az2, 21
az2_isnull, 21
az_isnull, 21
cond, 21
cx, 21
cx2, 21
cx2_isnull, 21
cx_isnull, 21
cy, 21
cy2, 22

cy2_isnull, 22
 cy_isnull, 22
 dsdir, 22
 dsdir_isnull, 22
 dsdist, 22
 dsdist2, 22
 dsdist2_isnull, 22
 dsdist_isnull, 22
 dsexp, 22
 dsexp2, 23
 dsexp2_isnull, 23
 dsexp_isnull, 23
 dshpid, 23
 dshpid_isnull, 23
 dskappa, 23
 dskappa2, 23
 dskappa2_isnull, 23
 dskappa_isnull, 23
 dsnrg, 23
 dsnrg2, 24
 dsnrg2_isnull, 24
 dsnrg_isnull, 24
 dsomega, 24
 dsomega2, 24
 dsomega2_isnull, 24
 dsomega_isnull, 24
 dsoscaxis, 24
 dsoscaxis2, 24
 dsoscaxis2_isnull, 24
 dsoscaxis_isnull, 25
 dsowidth, 25
 dsowidth2, 25
 dsowidth2_isnull, 25
 dsowidth_isnull, 25
 dsphi, 25
 dsphi2, 25
 dsphi2_isnull, 25
 dsphi_isnull, 25
 dspid, 25
 dspid_isnull, 25
 mutex, 26
 new_value_ready, 26
 no_rows_returned, 26
 sfn, 26
 sfn_isnull, 26
 sindex, 26
 sindex2, 26
 sindex2_isnull, 26
 sindex_isnull, 26
 skey, 26
 skey_isnull, 27
 sstart, 27
 sstart2, 27
 sstart2_isnull, 27
 sstart_isnull, 27
 stype, 27
 stype2, 27
 stype2_isnull, 27
 stype_isnull, 27
 lspg_nextshot_t
 pgpmac.h, 228
 lspg_nextshot_wait
 lspg.c, 105
 lspg_notice_processor
 kvredis.c, 59
 lspg.c, 105
 lspg_pg_connect
 kvredis.c, 60
 lspg.c, 105
 lspg_pg_service
 kvredis.c, 61
 lspg.c, 107
 lspg_query_next
 kvredis.c, 62
 lspg.c, 108
 lspg_query_push
 kvredis.c, 62
 lspg.c, 109
 lspg_query_queue
 kvredis.c, 67
 lspg.c, 118
 lspg_query_queue_off
 kvredis.c, 67
 lspg.c, 118
 lspg_query_queue_on
 kvredis.c, 68
 lspg.c, 118
 lspg_query_queue_reply
 kvredis.c, 68
 lspg.c, 118
 lspg_query_queue_t
 kvredis.c, 56
 lspg.c, 89
 lspg_query_reply_next
 kvredis.c, 63
 lspg.c, 109
 lspg_query_reply_peek
 kvredis.c, 63
 lspg.c, 109
 lspg_queue_cond
 lspg.c, 118
 lspg_queue_mutex
 lspg.c, 118
 lspg_receive
 kvredis.c, 64
 lspg.c, 110
 lspg_resetPoll_response
 kvredis.c, 68
 lspg.c, 118
 lspg_run
 lspg.c, 111
 pgpmac.h, 233
 lspg_scint_lut_cb
 lspg.c, 111
 lspg_send_next_query
 kvredis.c, 64

- lspg.c, [111](#)
- lspg_seq_run_prep
 - lspg.c, [118](#)
- lspg_seq_run_prep_all
 - lspg.c, [112](#)
 - pgpmac.h, [233](#)
- lspg_seq_run_prep_call
 - lspg.c, [112](#)
- lspg_seq_run_prep_cb
 - lspg.c, [113](#)
- lspg_seq_run_prep_done
 - lspg.c, [113](#)
- lspg_seq_run_prep_init
 - lspg.c, [113](#)
- lspg_seq_run_prep_struct, [28](#)
 - cond, [28](#)
 - mutex, [28](#)
 - new_value_ready, [28](#)
- lspg_seq_run_prep_t
 - lspg.c, [90](#)
- lspg_seq_run_prep_wait
 - lspg.c, [113](#)
- lspg_sig_service
 - lspg.c, [114](#)
- lspg_thread
 - lspg.c, [118](#)
- lspg_wait_for_detector
 - lspg.c, [119](#)
- lspg_wait_for_detector_all
 - lspg.c, [114](#)
- lspg_wait_for_detector_call
 - lspg.c, [114](#)
- lspg_wait_for_detector_cb
 - lspg.c, [114](#)
- lspg_wait_for_detector_done
 - lspg.c, [115](#)
- lspg_wait_for_detector_init
 - lspg.c, [115](#)
- lspg_wait_for_detector_struct, [28](#)
 - cond, [29](#)
 - mutex, [29](#)
 - new_value_ready, [29](#)
- lspg_wait_for_detector_t
 - lspg.c, [90](#)
- lspg_wait_for_detector_wait
 - lspg.c, [115](#)
- lspg_worker
 - lspg.c, [115](#)
- lspg_zoom_lut_call
 - pgpmac.h, [234](#)
- lspg_zoom_lut_cb
 - lspg.c, [116](#)
- lspgQueryQueueStruct, [29](#)
 - onResponse, [29](#)
 - qs, [29](#)
- lspgfd
 - kvredis.c, [68](#)
 - lspg.c, [119](#)
- lspmac.c, [119](#)
 - alignx, [171](#)
 - aligny, [171](#)
 - alignz, [171](#)
 - anal, [172](#)
 - apery, [172](#)
 - aperz, [172](#)
 - blight, [172](#)
 - blight_f, [172](#)
 - blight_ud, [172](#)
 - capy, [172](#)
 - capz, [172](#)
 - cenx, [172](#)
 - ceny, [172](#)
 - cleanstr, [130](#)
 - cr_cmd, [173](#)
 - cryo, [173](#)
 - cryo_switch, [173](#)
 - dbmem, [173](#)
 - dbmemIn, [173](#)
 - dryer, [173](#)
 - ethCmdOff, [173](#)
 - ethCmdOn, [173](#)
 - ethCmdQueue, [173](#)
 - ethCmdReply, [174](#)
 - flight, [174](#)
 - flight_f, [174](#)
 - flight_oo, [174](#)
 - fluo, [174](#)
 - fscint, [174](#)
 - fshut, [174](#)
 - gb_cmd, [174](#)
 - getivars, [174](#)
 - getmvars, [175](#)
 - hex_dump, [130](#)
 - kappa, [175](#)
 - LS_PMAC_STATE_CR, [126](#)
 - LS_PMAC_STATE_GB, [126](#)
 - LS_PMAC_STATE_GMR, [126](#)
 - LS_PMAC_STATE_IDLE, [127](#)
 - LS_PMAC_STATE_RR, [127](#)
 - LS_PMAC_STATE_SC, [127](#)
 - LS_PMAC_STATE_WACK, [127](#)
 - LS_PMAC_STATE_WCR, [127](#)
 - LS_PMAC_STATE_WGB, [127](#)
 - LSPMAC_PRESET_REGEX, [127](#)
 - linesReceived, [175](#)
 - ls_pmac_state, [175](#)
 - IsConnect, [130](#)
 - lspmac_Error, [134](#)
 - lspmac_GetAllIVars, [138](#)
 - lspmac_GetAllIVarsCB, [138](#)
 - lspmac_GetAllMVars, [139](#)
 - lspmac_GetAllMVarsCB, [139](#)
 - lspmac_GetShortReplyCB, [140](#)
 - lspmac_Getmem, [139](#)
 - lspmac_GetmemReplyCB, [139](#)
 - lspmac_Reset, [161](#)

- [lspmac_SendControlReplyPrintCB](#), 165
- [lspmac_Service](#), 165
- [lspmac_SockFlush](#), 168
- [lspmac_SockGetmem](#), 168
- [lspmac_SockSendControlCharPrint](#), 169
- [lspmac_SockSendline](#), 169
- [lspmac_SockSendline_nr](#), 169
- [lspmac_backLight_down_cb](#), 131
- [lspmac_backLight_up_cb](#), 132
- [lspmac_bi_init](#), 132
- [lspmac_bis](#), 175
- [lspmac_bo_init](#), 132
- [lspmac_bo_read](#), 133
- [lspmac_cryoSwitchChanged_cb](#), 133
- [lspmac_dac_init](#), 133
- [lspmac_dac_read](#), 134
- [lspmac_fshut_init](#), 135
- [lspmac_get_status](#), 135
- [lspmac_get_status_cb](#), 136
- [lspmac_getPosition](#), 140
- [lspmac_home1_queue](#), 141
- [lspmac_home2_queue](#), 142
- [lspmac_init](#), 142
- [lspmac_jogabs_queue](#), 144
- [lspmac_light_zoom_cb](#), 145
- [lspmac_lut](#), 145
- [lspmac_motor_init](#), 146
- [lspmac_motors](#), 175
- [lspmac_move_or_jog_abs_queue](#), 147
- [lspmac_move_or_jog_preset_queue](#), 148
- [lspmac_move_preset_queue](#), 149
- [lspmac_moveabs_blight_factor_queue](#), 150
- [lspmac_moveabs_bo_queue](#), 150
- [lspmac_moveabs_flight_factor_queue](#), 151
- [lspmac_moveabs_frontlight_oo_queue](#), 151
- [lspmac_moveabs_fshut_queue](#), 151
- [lspmac_moveabs_queue](#), 152
- [lspmac_moveabs_timed_queue](#), 152
- [lspmac_moveabs_wait](#), 153
- [lspmac_movedac_queue](#), 154
- [lspmac_movezoom_queue](#), 155
- [lspmac_moving_cond](#), 175
- [lspmac_moving_flags](#), 175
- [lspmac_moving_mutex](#), 175
- [lspmac_nbis](#), 176
- [lspmac_newKV_cb](#), 155
- [lspmac_next_state](#), 156
- [lspmac_nmotors](#), 176
- [lspmac_pmacmotor_read](#), 157
- [lspmac_pop_queue](#), 160
- [lspmac_pop_reply](#), 160
- [lspmac_push_queue](#), 161
- [lspmac_rlut](#), 161
- [lspmac_run](#), 162
- [lspmac_scint_dried_cb](#), 162
- [lspmac_scint_inPosition_cb](#), 163
- [lspmac_send_command](#), 163
- [lspmac_sendcmd](#), 164
- [lspmac_sendcmd_nocb](#), 164
- [lspmac_shutter_cond](#), 176
- [lspmac_shutter_has_opened](#), 176
- [lspmac_shutter_mutex](#), 176
- [lspmac_shutter_read](#), 167
- [lspmac_shutter_state](#), 176
- [lspmac_soft_motor_init](#), 170
- [lspmac_soft_motor_read](#), 170
- [lspmac_status_last_time](#), 176
- [lspmac_status_time](#), 176
- [lspmac_video_rotate](#), 170
- [lspmac_worker](#), 171
- [md2_status](#), 176
- [md2_status_mutex](#), 177
- [md2_status_t](#), 130
- [now](#), 177
- [omega](#), 177
- [omega_zero_search](#), 177
- [omega_zero_time](#), 177
- [omega_zero_velocity](#), 177
- [PMAC_MIN_CMD_TIME](#), 128
- [PMACPORT](#), 128
- [phi](#), 177
- [pmac_cmd_size](#), 128
- [pmac_error_strs](#), 177
- [pmac_queue_cond](#), 178
- [pmac_queue_mutex](#), 178
- [pmac_thread](#), 178
- [pmacfd](#), 178
- [rr_cmd](#), 178
- [scint](#), 178
- [VR_CTRL_RESPONSE](#), 128
- [VR_DOWNLOAD](#), 128
- [VR_FWDOWNLOAD](#), 128
- [VR_IPADDRESS](#), 128
- [VR_PMAC_FLUSH](#), 128
- [VR_PMAC_GETBUFFER](#), 128
- [VR_PMAC_GETLINE](#), 128
- [VR_PMAC_GETMEM](#), 128
- [VR_PMAC_GETRESPONSE](#), 129
- [VR_PMAC_PORT](#), 129
- [VR_PMAC_READREADY](#), 129
- [VR_PMAC_SENDLINE](#), 129
- [VR_PMAC_SETBIT](#), 129
- [VR_PMAC_SETBITS](#), 129
- [VR_PMAC_SETMEM](#), 129
- [VR_PMAC_WRITEBUFFER](#), 129
- [VR_PMAC_WRITEERROR](#), 129
- [VR_UPLOAD](#), 129
- [zoom](#), 178
- [lspmac_Error](#)
 - [lspmac.c](#), 134
- [lspmac_GetAllIVars](#)
 - [lspmac.c](#), 138
- [lspmac_GetAllIVarsCB](#)
 - [lspmac.c](#), 138
- [lspmac_GetAllMVars](#)
 - [lspmac.c](#), 139

lspmac_GetAllMVarsCB
 lspmac.c, [139](#)
 lspmac_GetShortReplyCB
 lspmac.c, [140](#)
 lspmac_Getmem
 lspmac.c, [139](#)
 lspmac_GetmemReplyCB
 lspmac.c, [139](#)
 lspmac_Reset
 lspmac.c, [161](#)
 lspmac_SendControlReplyPrintCB
 lspmac.c, [165](#)
 lspmac_Service
 lspmac.c, [165](#)
 lspmac_SockFlush
 lspmac.c, [168](#)
 lspmac_SockGetmem
 lspmac.c, [168](#)
 lspmac_SockSendControlCharPrint
 lspmac.c, [169](#)
 lspmac_SockSendline
 lspmac.c, [169](#)
 pgpmac.h, [237](#)
 lspmac_SockSendline_nr
 lspmac.c, [169](#)
 lspmac_backLight_down_cb
 lspmac.c, [131](#)
 lspmac_backLight_up_cb
 lspmac.c, [132](#)
 lspmac_bi_init
 lspmac.c, [132](#)
 lspmac_bi_struct, [30](#)
 changeEventOff, [30](#)
 changeEventOn, [30](#)
 first_time, [30](#)
 mask, [30](#)
 mutex, [31](#)
 previous, [31](#)
 ptr, [31](#)
 lspmac_bi_t
 pgpmac.h, [228](#)
 lspmac_bis
 lspmac.c, [175](#)
 lspmac_bo_init
 lspmac.c, [132](#)
 lspmac_bo_read
 lspmac.c, [133](#)
 lspmac_cmd_queue_struct, [31](#)
 no_reply, [32](#)
 onResponse, [32](#)
 pcmd, [32](#)
 rbuff, [32](#)
 time_sent, [32](#)
 lspmac_cryoSwitchChanged_cb
 lspmac.c, [133](#)
 lspmac_dac_init
 lspmac.c, [133](#)
 lspmac_dac_read
 lspmac.c, [134](#)
 lspmac_fshut_init
 lspmac.c, [135](#)
 lspmac_get_status
 lspmac.c, [135](#)
 lspmac_get_status_cb
 lspmac.c, [136](#)
 lspmac_getPosition
 lspmac.c, [140](#)
 pgpmac.h, [234](#)
 lspmac_home1_queue
 lspmac.c, [141](#)
 lspmac_home2_queue
 lspmac.c, [142](#)
 lspmac_init
 lspmac.c, [142](#)
 pgpmac.h, [234](#)
 lspmac_jogabs_queue
 lspmac.c, [144](#)
 pgpmac.h, [236](#)
 lspmac_light_zoom_cb
 lspmac.c, [145](#)
 lspmac_lut
 lspmac.c, [145](#)
 lspmac_motor_init
 lspmac.c, [146](#)
 lspmac_motor_struct, [32](#)
 actual_pos_cnts, [34](#)
 actual_pos_cnts_p, [34](#)
 axis, [34](#)
 cond, [34](#)
 coord_num, [34](#)
 dac_mvar, [35](#)
 format, [35](#)
 home, [35](#)
 homing, [35](#)
 lspg_initialized, [35](#)
 lut, [35](#)
 max_accel, [35](#)
 max_speed, [35](#)
 motion_seen, [35](#)
 motor_num, [36](#)
 moveAbs, [36](#)
 mutex, [36](#)
 name, [36](#)
 nlut, [36](#)
 not_done, [36](#)
 position, [36](#)
 pq, [36](#)
 preset_regex, [36](#)
 presets, [37](#)
 read, [37](#)
 read_mask, [37](#)
 read_ptr, [37](#)
 reported_position, [37](#)
 requested_pos_cnts, [37](#)
 requested_position, [37](#)
 status1, [37](#)

- status1_p, [37](#)
- status2, [38](#)
- status2_p, [38](#)
- statuss, [38](#)
- u2c, [38](#)
- units, [38](#)
- update_format, [38](#)
- update_resolution, [38](#)
- win, [38](#)
- write_fmt, [38](#)
- lspmac_motor_t
 - pgpmac.h, [228](#)
- lspmac_motors
 - lspmac.c, [175](#)
 - pgpmac.h, [244](#)
- lspmac_move_or_jog_abs_queue
 - lspmac.c, [147](#)
- lspmac_move_or_jog_preset_queue
 - lspmac.c, [148](#)
 - pgpmac.h, [236](#)
- lspmac_move_or_jog_queue
 - pgpmac.h, [236](#)
- lspmac_move_preset_queue
 - lspmac.c, [149](#)
- lspmac_moveabs_blight_factor_queue
 - lspmac.c, [150](#)
- lspmac_moveabs_bo_queue
 - lspmac.c, [150](#)
- lspmac_moveabs_flight_factor_queue
 - lspmac.c, [151](#)
- lspmac_moveabs_frontlight_oo_queue
 - lspmac.c, [151](#)
- lspmac_moveabs_fshut_queue
 - lspmac.c, [151](#)
- lspmac_moveabs_queue
 - lspmac.c, [152](#)
 - pgpmac.h, [236](#)
- lspmac_moveabs_timed_queue
 - lspmac.c, [152](#)
- lspmac_moveabs_wait
 - lspmac.c, [153](#)
- lspmac_movedac_queue
 - lspmac.c, [154](#)
- lspmac_movezoom_queue
 - lspmac.c, [155](#)
- lspmac_moving_cond
 - lspmac.c, [175](#)
 - pgpmac.h, [244](#)
- lspmac_moving_flags
 - lspmac.c, [175](#)
 - pgpmac.h, [244](#)
- lspmac_moving_mutex
 - lspmac.c, [175](#)
 - pgpmac.h, [245](#)
- lspmac_nbis
 - lspmac.c, [176](#)
- lspmac_newKV_cb
 - lspmac.c, [155](#)
- lspmac_next_state
 - lspmac.c, [156](#)
- lspmac_nmotors
 - lspmac.c, [176](#)
 - pgpmac.h, [245](#)
- lspmac_pmacmotor_read
 - lspmac.c, [157](#)
- lspmac_pop_queue
 - lspmac.c, [160](#)
- lspmac_pop_reply
 - lspmac.c, [160](#)
- lspmac_push_queue
 - lspmac.c, [161](#)
- lspmac_rlut
 - lspmac.c, [161](#)
- lspmac_run
 - lspmac.c, [162](#)
 - pgpmac.h, [236](#)
- lspmac_scint_dried_cb
 - lspmac.c, [162](#)
- lspmac_scint_inPosition_cb
 - lspmac.c, [163](#)
- lspmac_send_command
 - lspmac.c, [163](#)
- lspmac_sendcmd
 - lspmac.c, [164](#)
- lspmac_sendcmd_nocb
 - lspmac.c, [164](#)
- lspmac_shutter_cond
 - lspmac.c, [176](#)
 - pgpmac.h, [245](#)
- lspmac_shutter_has_opened
 - lspmac.c, [176](#)
 - pgpmac.h, [245](#)
- lspmac_shutter_mutex
 - lspmac.c, [176](#)
 - pgpmac.h, [245](#)
- lspmac_shutter_read
 - lspmac.c, [167](#)
- lspmac_shutter_state
 - lspmac.c, [176](#)
 - pgpmac.h, [245](#)
- lspmac_soft_motor_init
 - lspmac.c, [170](#)
- lspmac_soft_motor_read
 - lspmac.c, [170](#)
- lspmac_status_last_time
 - lspmac.c, [176](#)
- lspmac_status_time
 - lspmac.c, [176](#)
- lspmac_video_rotate
 - lspmac.c, [170](#)
- lspmac_worker
 - lspmac.c, [171](#)
- lsredis.c, [179](#)
 - _lsredis_get_obj, [180](#)
 - _lsredis_set_value, [181](#)
 - lsredis_addRead, [182](#)

- lsredis_addWrite, 182
- lsredis_cleanup, 182
- lsredis_debugCB, 182
- lsredis_delRead, 183
- lsredis_delWrite, 183
- lsredis_fd_service, 183
- lsredis_get_obj, 184
- lsredis_getd, 184
- lsredis_getl, 185
- lsredis_getstr, 185
- lsredis_head, 192
- lsredis_hgetCB, 185
- lsredis_init, 185
- lsredis_isvalid, 186
- lsredis_key_select_regex, 192
- lsredis_keysCB, 187
- lsredis_maybe_add_key, 187
- lsredis_objs, 192
- lsredis_objs_mutex, 192
- lsredis_publisher, 192
- lsredis_ro_mutex, 192
- lsredis_run, 187
- lsredis_select, 187
- lsredis_set_invalid, 188
- lsredis_set_value, 188
- lsredis_setstr, 188
- lsredis_subCB, 189
- lsredis_thread, 192
- lsredis_worker, 191
- lsredis_wr_mutex, 192
- redisDisconnectCB, 191
- roac, 192
- rofd, 192
- subac, 193
- subfd, 193
- wrac, 193
- wrfd, 193
- lsredis_addRead
 - lsredis.c, 182
- lsredis_addWrite
 - lsredis.c, 182
- lsredis_cleanup
 - lsredis.c, 182
- lsredis_debugCB
 - lsredis.c, 182
- lsredis_delRead
 - lsredis.c, 183
- lsredis_delWrite
 - lsredis.c, 183
- lsredis_fd_service
 - lsredis.c, 183
- lsredis_get_obj
 - lsredis.c, 184
 - pgpmac.h, 237
- lsredis_getd
 - lsredis.c, 184
- lsredis_getl
 - lsredis.c, 185
- lsredis_getstr
 - lsredis.c, 185
- lsredis_head
 - lsredis.c, 192
- lsredis_hgetCB
 - lsredis.c, 185
- lsredis_init
 - lsredis.c, 185
 - pgpmac.h, 238
- lsredis_isvalid
 - lsredis.c, 186
- lsredis_key_select_regex
 - lsredis.c, 192
- lsredis_keysCB
 - lsredis.c, 187
- lsredis_maybe_add_key
 - lsredis.c, 187
- lsredis_obj_struct, 39
 - cond, 40
 - events_name, 40
 - getd, 40
 - getl, 40
 - getstr, 40
 - hits, 40
 - key, 40
 - mutex, 40
 - next, 40
 - valid, 40
 - value, 41
 - value_length, 41
 - wait_for_me, 41
- lsredis_obj_t
 - pgpmac.h, 228
- lsredis_objs
 - lsredis.c, 192
- lsredis_objs_mutex
 - lsredis.c, 192
- lsredis_publisher
 - lsredis.c, 192
- lsredis_ro_mutex
 - lsredis.c, 192
- lsredis_run
 - lsredis.c, 187
 - pgpmac.h, 239
- lsredis_select
 - lsredis.c, 187
- lsredis_set_invalid
 - lsredis.c, 188
- lsredis_set_value
 - lsredis.c, 188
- lsredis_setstr
 - lsredis.c, 188
- lsredis_subCB
 - lsredis.c, 189
- lsredis_thread
 - lsredis.c, 192
- lsredis_worker
 - lsredis.c, 191

- lsredis_wr_mutex
 - lsredis.c, 192
- lstimer.c, 193
 - handler, 195
 - LSTIMER_LIST_LENGTH, 194
 - lstimer_active_timers, 198
 - lstimer_add_timer, 195
 - lstimer_cond, 199
 - lstimer_init, 196
 - lstimer_list, 199
 - lstimer_list_t, 195
 - lstimer_mutex, 199
 - lstimer_run, 196
 - lstimer_thread, 199
 - lstimer_timerid, 199
 - lstimer_worker, 196
 - new_timer, 199
 - service_timers, 197
- lstimer_active_timers
 - lstimer.c, 198
- lstimer_add_timer
 - lstimer.c, 195
 - pgpmac.h, 239
- lstimer_cond
 - lstimer.c, 199
- lstimer_init
 - lstimer.c, 196
 - pgpmac.h, 239
- lstimer_list
 - lstimer.c, 199
- lstimer_list_struct, 41
 - delay_nsecs, 42
 - delay_secs, 42
 - event, 42
 - init_nsecs, 42
 - init_secs, 42
 - last_nsecs, 42
 - last_secs, 42
 - ncalls, 42
 - next_nsecs, 43
 - next_secs, 43
 - shots, 43
- lstimer_list_t
 - lstimer.c, 195
- lstimer_mutex
 - lstimer.c, 199
- lstimer_run
 - lstimer.c, 196
 - pgpmac.h, 240
- lstimer_thread
 - lstimer.c, 199
- lstimer_timerid
 - lstimer.c, 199
- lstimer_worker
 - lstimer.c, 196
- lsupdate.c, 199
 - lsupdate_init, 200
 - lsupdate_run, 200
 - lsupdate_thread, 202
 - lsupdate_updateit, 200
 - lsupdate_worker, 201
- lsupdate_init
 - lsupdate.c, 200
 - pgpmac.h, 240
- lsupdate_run
 - lsupdate.c, 200
 - pgpmac.h, 240
- lsupdate_thread
 - lsupdate.c, 202
- lsupdate_updateit
 - lsupdate.c, 200
- lsupdate_worker
 - lsupdate.c, 201
- ltime
 - lslogging_queue_struct, 13
- lut
 - lspmac_motor_struct, 35
- MD2CMDS_CMD_LENGTH
 - pgpmac.h, 227
- main
 - kvredis.c, 65
 - pgpmac.c, 218
- mask
 - lspmac_bi_struct, 30
- max_accel
 - lspmac_motor_struct, 35
- max_speed
 - lspmac_motor_struct, 35
- md2_status
 - lspmac.c, 176
- md2_status_mutex
 - lspmac.c, 177
 - pgpmac.h, 245
- md2_status_t
 - lspmac.c, 130
- md2StatusStruct, 43
 - acc11c_1, 45
 - acc11c_2, 45
 - acc11c_3, 45
 - acc11c_5, 45
 - acc11c_6, 45
 - alignx_act_pos, 45
 - alignx_status_1, 45
 - alignx_status_2, 45
 - aligny_act_pos, 45
 - aligny_status_1, 45
 - aligny_status_2, 45
 - alignz_act_pos, 45
 - alignz_status_1, 45
 - alignz_status_2, 46
 - analyzer_act_pos, 46
 - analyzer_status_1, 46
 - analyzer_status_2, 46
 - aperturey_act_pos, 46
 - aperturey_status_1, 46
 - aperturey_status_2, 46

- aperturez_act_pos, [46](#)
- aperturez_status_1, [46](#)
- aperturez_status_2, [46](#)
- back_dac, [46](#)
- capy_act_pos, [46](#)
- capy_status_1, [47](#)
- capy_status_2, [47](#)
- capz_act_pos, [47](#)
- capz_status_1, [47](#)
- capz_status_2, [47](#)
- centerx_act_pos, [47](#)
- centerx_status_1, [47](#)
- centerx_status_2, [47](#)
- centery_act_pos, [47](#)
- centery_status_1, [47](#)
- centery_status_2, [47](#)
- dummy1, [47](#)
- dummy2, [48](#)
- dummy3, [48](#)
- dummy4, [48](#)
- dummy5, [48](#)
- dummy6, [48](#)
- dummy7, [48](#)
- dummy8, [48](#)
- dummy9, [48](#)
- dummyA, [48](#)
- dummyB, [48](#)
- front_dac, [48](#)
- fs_has_opened, [48](#)
- fs_has_opened_globally, [49](#)
- fs_is_open, [49](#)
- kappa_act_pos, [49](#)
- kappa_status_1, [49](#)
- kappa_status_2, [49](#)
- moving_flags, [49](#)
- number_passes, [49](#)
- omega_act_pos, [49](#)
- omega_status_1, [49](#)
- omega_status_2, [49](#)
- phi_act_pos, [49](#)
- phi_status_1, [49](#)
- phi_status_2, [50](#)
- phiscan, [50](#)
- scint_act_pos, [50](#)
- scint_piezo, [50](#)
- scint_status_1, [50](#)
- scint_status_2, [50](#)
- zoom_act_pos, [50](#)
- zoom_status_1, [50](#)
- zoom_status_2, [50](#)
- md2cmds.c, [202](#)
 - md2cmds_center, [204](#)
 - md2cmds_cmd, [216](#)
 - md2cmds_collect, [204](#)
 - md2cmds_cond, [216](#)
 - md2cmds_init, [206](#)
 - md2cmds_maybe_done_moving_cb, [206](#)
 - md2cmds_maybe_rotate_done_cb, [207](#)
 - md2cmds_moveAbs, [207](#)
 - md2cmds_moving_cond, [216](#)
 - md2cmds_moving_count, [216](#)
 - md2cmds_moving_mutex, [216](#)
 - md2cmds_moving_pq, [216](#)
 - md2cmds_mutex, [216](#)
 - md2cmds_mvcenter_move, [208](#)
 - md2cmds_mvcenter_prep, [209](#)
 - md2cmds_mvcenter_wait, [210](#)
 - md2cmds_phase_change, [210](#)
 - md2cmds_prep_motion, [212](#)
 - md2cmds_rotate, [212](#)
 - md2cmds_rotate_cb, [214](#)
 - md2cmds_run, [214](#)
 - md2cmds_set_scale_cb, [215](#)
 - md2cmds_thread, [216](#)
 - md2cmds_transfer, [215](#)
 - md2cmds_worker, [215](#)
 - rotating, [217](#)
- md2cmds_center
 - md2cmds.c, [204](#)
- md2cmds_cmd
 - md2cmds.c, [216](#)
 - pgpmac.h, [245](#)
- md2cmds_collect
 - md2cmds.c, [204](#)
- md2cmds_cond
 - md2cmds.c, [216](#)
 - pgpmac.h, [245](#)
- md2cmds_init
 - md2cmds.c, [206](#)
 - pgpmac.h, [240](#)
- md2cmds_maybe_done_moving_cb
 - md2cmds.c, [206](#)
- md2cmds_maybe_rotate_done_cb
 - md2cmds.c, [207](#)
- md2cmds_moveAbs
 - md2cmds.c, [207](#)
- md2cmds_moving_cond
 - md2cmds.c, [216](#)
- md2cmds_moving_count
 - md2cmds.c, [216](#)
- md2cmds_moving_mutex
 - md2cmds.c, [216](#)
- md2cmds_moving_pq
 - md2cmds.c, [216](#)
- md2cmds_mutex
 - md2cmds.c, [216](#)
 - pgpmac.h, [246](#)
- md2cmds_mvcenter_move
 - md2cmds.c, [208](#)
- md2cmds_mvcenter_prep
 - md2cmds.c, [209](#)
- md2cmds_mvcenter_wait
 - md2cmds.c, [210](#)
- md2cmds_pg_cond
 - pgpmac.h, [246](#)
- md2cmds_pg_mutex

- pgpmac.h, 246
- md2cmds_phase_change
 - md2cmds.c, 210
- md2cmds_prep_motion
 - md2cmds.c, 212
- md2cmds_rotate
 - md2cmds.c, 212
- md2cmds_rotate_cb
 - md2cmds.c, 214
- md2cmds_run
 - md2cmds.c, 214
 - pgpmac.h, 240
- md2cmds_set_scale_cb
 - md2cmds.c, 215
- md2cmds_thread
 - md2cmds.c, 216
- md2cmds_transfer
 - md2cmds.c, 215
- md2cmds_worker
 - md2cmds.c, 215
- motion_seen
 - lspmac_motor_struct, 35
- motor_num
 - lspmac_motor_struct, 36
- moveAbs
 - lspmac_motor_struct, 36
- moving_flags
 - md2StatusStruct, 49
- mutex
 - lspg_getcenter_struct, 15
 - lspg_lock_detector_struct, 16
 - lspg_lock_diffractionmeter_struct, 16
 - lspg_nextshot_struct, 26
 - lspg_seq_run_prep_struct, 28
 - lspg_wait_for_detector_struct, 29
 - lspmac_bi_struct, 31
 - lspmac_motor_struct, 36
 - lsredis_obj_struct, 40
- name
 - lspmac_motor_struct, 36
- ncalls
 - lstimer_list_struct, 42
- ncurses_mutex
 - pgpmac.c, 221
 - pgpmac.h, 246
- new_timer
 - lstimer.c, 199
- new_value_ready
 - lspg_getcenter_struct, 15
 - lspg_lock_detector_struct, 16
 - lspg_lock_diffractionmeter_struct, 17
 - lspg_nextshot_struct, 26
 - lspg_seq_run_prep_struct, 28
 - lspg_wait_for_detector_struct, 29
- next
 - lsevents_listener_struct, 9
 - lskvs_kvs_list_struct, 11
 - lskvs_kvs_struct, 12
 - lsredis_obj_struct, 40
- next_nsecs
 - lstimer_list_struct, 43
- next_secs
 - lstimer_list_struct, 43
- nlut
 - lspmac_motor_struct, 36
- no_reply
 - lspmac_cmd_queue_struct, 32
- no_rows_returned
 - lspg_getcenter_struct, 15
 - lspg_nextshot_struct, 26
- not_done
 - lspmac_motor_struct, 36
- now
 - kvredis.c, 68
 - lspg.c, 119
 - lspmac.c, 177
- number_passes
 - md2StatusStruct, 49
- omega
 - lspmac.c, 177
 - pgpmac.h, 246
- omega_act_pos
 - md2StatusStruct, 49
- omega_status_1
 - md2StatusStruct, 49
- omega_status_2
 - md2StatusStruct, 49
- omega_zero_search
 - lspmac.c, 177
- omega_zero_time
 - lspmac.c, 177
 - pgpmac.h, 246
- omega_zero_velocity
 - lspmac.c, 177
- onResponse
 - lspgQueryQueueStruct, 29
 - lspmac_cmd_queue_struct, 32
- PMAC_MIN_CMD_TIME
 - lspmac.c, 128
- PMACPORT
 - lspmac.c, 128
- pcmd
 - lspmac_cmd_queue_struct, 32
- pgpmac.c, 217
 - main, 218
 - ncurses_mutex, 221
 - pgpmac_printf, 219
 - stdinService, 220
 - stdinfda, 221
 - term_input, 221
 - term_output, 221
 - term_status, 221
 - term_status2, 221
- pgpmac.h, 222
 - alignx, 241

[aligny](#), 242
[alignz](#), 242
[anal](#), 242
[apery](#), 242
[aperz](#), 242
[blight](#), 242
[blight_f](#), 242
[blight_ud](#), 242
[capy](#), 242
[capz](#), 242
[cenx](#), 243
[ceny](#), 243
[cryo](#), 243
[dryer](#), 243
[flight](#), 243
[flight_f](#), 243
[flight_oo](#), 243
[fluo](#), 243
[fscint](#), 243
[fshut](#), 244
[kappa](#), 244
[lsevents_add_listener](#), 228
[lsevents_init](#), 229
[lsevents_remove_listener](#), 229
[lsevents_run](#), 230
[lsevents_send_event](#), 230
[lskvs_find_preset_position](#), 231
[lskvs_kvs](#), 244
[lskvs_kvs_list_t](#), 227
[lskvs_kvs_t](#), 227
[lskvs_regcomp](#), 232
[lskvs_rwlock](#), 244
[lspg_getcenter](#), 244
[lspg_getcenter_t](#), 228
[lspg_init](#), 233
[lspg_nextshot](#), 244
[lspg_nextshot_t](#), 228
[lspg_run](#), 233
[lspg_seq_run_prep_all](#), 233
[lspg_zoom_lut_call](#), 234
[lspmac_SockSendline](#), 237
[lspmac_bi_t](#), 228
[lspmac_getPosition](#), 234
[lspmac_init](#), 234
[lspmac_jogabs_queue](#), 236
[lspmac_motor_t](#), 228
[lspmac_motors](#), 244
[lspmac_move_or_jog_preset_queue](#), 236
[lspmac_move_or_jog_queue](#), 236
[lspmac_moveabs_queue](#), 236
[lspmac_moving_cond](#), 244
[lspmac_moving_flags](#), 244
[lspmac_moving_mutex](#), 245
[lspmac_nmotors](#), 245
[lspmac_run](#), 236
[lspmac_shutter_cond](#), 245
[lspmac_shutter_has_opened](#), 245
[lspmac_shutter_mutex](#), 245
[lspmac_shutter_state](#), 245
[lsredis_get_obj](#), 237
[lsredis_init](#), 238
[lsredis_obj_t](#), 228
[lsredis_run](#), 239
[lster_timer_add_timer](#), 239
[lster_timer_init](#), 239
[lster_timer_run](#), 240
[lsupdate_init](#), 240
[lsupdate_run](#), 240
[MD2CMDS_CMD_LENGTH](#), 227
[md2_status_mutex](#), 245
[md2cmds_cmd](#), 245
[md2cmds_cond](#), 245
[md2cmds_init](#), 240
[md2cmds_mutex](#), 246
[md2cmds_pg_cond](#), 246
[md2cmds_pg_mutex](#), 246
[md2cmds_run](#), 240
[ncurses_mutex](#), 246
[omega](#), 246
[omega_zero_time](#), 246
[pgpmac_printf](#), 241
[phi](#), 246
[pmac_cmd_queue_t](#), 228
[pmac_cmd_t](#), 228
[pmac_queue_cond](#), 246
[pmac_queue_mutex](#), 246
[PmacSockSendline](#), 241
[scint](#), 246
[term_input](#), 246
[term_output](#), 247
[term_status](#), 247
[term_status2](#), 247
[zoom](#), 247
[pgpmac_printf](#)
[pgpmac.c](#), 219
[pgpmac.h](#), 241
[phi](#)
[lspmac.c](#), 177
[pgpmac.h](#), 246
[phi_act_pos](#)
[md2StatusStruct](#), 49
[phi_status_1](#)
[md2StatusStruct](#), 49
[phi_status_2](#)
[md2StatusStruct](#), 50
[phiscan](#)
[md2StatusStruct](#), 50
[pmac_cmd_queue_t](#)
[pgpmac.h](#), 228
[pmac_cmd_size](#)
[lspmac.c](#), 128
[pmac_cmd_t](#)
[pgpmac.h](#), 228
[pmac_error_strs](#)
[lspmac.c](#), 177
[pmac_queue_cond](#)

- lspmac.c, [178](#)
- pgpmac.h, [246](#)
- pmac_queue_mutex
 - lspmac.c, [178](#)
 - pgpmac.h, [246](#)
- pmac_thread
 - lspmac.c, [178](#)
- PmacSockSendline
 - pgpmac.h, [241](#)
- pmacfd
 - lspmac.c, [178](#)
- position
 - lspmac_motor_struct, [36](#)
- pq
 - lspmac_motor_struct, [36](#)
- preset_regex
 - lspmac_motor_struct, [36](#)
- presets
 - lspmac_motor_struct, [37](#)
- previous
 - lspmac_bi_struct, [31](#)
- ptr
 - lspmac_bi_struct, [31](#)
- q
 - kvredis.c, [68](#)
 - lspg.c, [119](#)
- qs
 - lspgQueryQueueStruct, [29](#)
- raw_regexp
 - lsevents_listener_struct, [9](#)
- rbuff
 - lspmac_cmd_queue_struct, [32](#)
- re
 - lsevents_listener_struct, [10](#)
- read
 - lspmac_motor_struct, [37](#)
- read_mask
 - lspmac_motor_struct, [37](#)
- read_ptr
 - lspmac_motor_struct, [37](#)
- redisDisconnectCB
 - kvredis.c, [67](#)
 - lsredis.c, [191](#)
- reported_position
 - lspmac_motor_struct, [37](#)
- Request
 - tagEthernetCmd, [51](#)
- RequestType
 - tagEthernetCmd, [51](#)
- requested_pos_cnts
 - lspmac_motor_struct, [37](#)
- requested_position
 - lspmac_motor_struct, [37](#)
- roac
 - lsredis.c, [192](#)
- rofd
 - lsredis.c, [192](#)
- rotating
 - md2cmds.c, [217](#)
- rr_cmd
 - lspmac.c, [178](#)
- scint
 - lspmac.c, [178](#)
 - pgpmac.h, [246](#)
- scint_act_pos
 - md2StatusStruct, [50](#)
- scint_piezo
 - md2StatusStruct, [50](#)
- scint_status_1
 - md2StatusStruct, [50](#)
- scint_status_2
 - md2StatusStruct, [50](#)
- service_timers
 - lstimer.c, [197](#)
- sfn
 - lspg_nextshot_struct, [26](#)
- sfn_isnull
 - lspg_nextshot_struct, [26](#)
- shots
 - lstimer_list_struct, [43](#)
- sindex
 - lspg_nextshot_struct, [26](#)
- sindex2
 - lspg_nextshot_struct, [26](#)
- sindex2_isnull
 - lspg_nextshot_struct, [26](#)
- sindex_isnull
 - lspg_nextshot_struct, [26](#)
- skey
 - lspg_nextshot_struct, [26](#)
- skey_isnull
 - lspg_nextshot_struct, [27](#)
- sstart
 - lspg_nextshot_struct, [27](#)
- sstart2
 - lspg_nextshot_struct, [27](#)
- sstart2_isnull
 - lspg_nextshot_struct, [27](#)
- sstart_isnull
 - lspg_nextshot_struct, [27](#)
- status1
 - lspmac_motor_struct, [37](#)
- status1_p
 - lspmac_motor_struct, [37](#)
- status2
 - lspmac_motor_struct, [38](#)
- status2_p
 - lspmac_motor_struct, [38](#)
- statuss
 - lspmac_motor_struct, [38](#)
- stdinService
 - pgpmac.c, [220](#)
- stdinfda
 - pgpmac.c, [221](#)
- stype

- lspg_nextshot_struct, [27](#)
- stype2
 - lspg_nextshot_struct, [27](#)
- stype2_isnull
 - lspg_nextshot_struct, [27](#)
- stype_isnull
 - lspg_nextshot_struct, [27](#)
- subac
 - kvredis.c, [68](#)
 - lsredis.c, [193](#)
- subfd
 - kvredis.c, [68](#)
 - lsredis.c, [193](#)
- tagEthernetCmd, [50](#)
 - bData, [51](#)
 - Request, [51](#)
 - RequestType, [51](#)
 - wIndex, [51](#)
 - wLength, [51](#)
 - wValue, [51](#)
- term_input
 - pgpmac.c, [221](#)
 - pgpmac.h, [246](#)
- term_output
 - pgpmac.c, [221](#)
 - pgpmac.h, [247](#)
- term_status
 - pgpmac.c, [221](#)
 - pgpmac.h, [247](#)
- term_status2
 - pgpmac.c, [221](#)
 - pgpmac.h, [247](#)
- time_sent
 - lspmac_cmd_queue_struct, [32](#)
- u2c
 - lspmac_motor_struct, [38](#)
- units
 - lspmac_motor_struct, [38](#)
- update_format
 - lspmac_motor_struct, [38](#)
- update_resolution
 - lspmac_motor_struct, [38](#)
- v
 - lskvs_kvs_struct, [12](#)
- VR_CTRL_RESPONSE
 - lspmac.c, [128](#)
- VR_DOWNLOAD
 - lspmac.c, [128](#)
- VR_FWDOWNLOAD
 - lspmac.c, [128](#)
- VR_IPADDRESS
 - lspmac.c, [128](#)
- VR_PMAC_FLUSH
 - lspmac.c, [128](#)
- VR_PMAC_GETBUFFER
 - lspmac.c, [128](#)
- VR_PMAC_GETLINE
 - lspmac.c, [128](#)
- VR_PMAC_GETMEM
 - lspmac.c, [128](#)
- VR_PMAC_GETRESPONSE
 - lspmac.c, [129](#)
- VR_PMAC_PORT
 - lspmac.c, [129](#)
- VR_PMAC_READREADY
 - lspmac.c, [129](#)
- VR_PMAC_SENDCTRLCHAR
 - lspmac.c, [129](#)
- VR_PMAC_SENDLINE
 - lspmac.c, [129](#)
- VR_PMAC_SETBIT
 - lspmac.c, [129](#)
- VR_PMAC_SETBITS
 - lspmac.c, [129](#)
- VR_PMAC_SETMEM
 - lspmac.c, [129](#)
- VR_PMAC_WRITEBUFFER
 - lspmac.c, [129](#)
- VR_PMAC_WRITEERROR
 - lspmac.c, [129](#)
- VR_UPLOAD
 - lspmac.c, [129](#)
- valid
 - lsredis_obj_struct, [40](#)
- value
 - lsredis_obj_struct, [41](#)
- value_length
 - lsredis_obj_struct, [41](#)
- vl
 - lskvs_kvs_struct, [12](#)
- wIndex
 - tagEthernetCmd, [51](#)
- wLength
 - tagEthernetCmd, [51](#)
- wValue
 - tagEthernetCmd, [51](#)
- wait_for_me
 - lsredis_obj_struct, [41](#)
- win
 - lspmac_motor_struct, [38](#)
- wrac
 - lsredis.c, [193](#)
- wrfd
 - lsredis.c, [193](#)
- write_fmt
 - lspmac_motor_struct, [38](#)
- zoom
 - lspg_getcenter_struct, [15](#)
 - lspmac.c, [178](#)
 - pgpmac.h, [247](#)
- zoom_act_pos
 - md2StatusStruct, [50](#)
- zoom_isnull

lspg_getcenter_struct, [15](#)
zoom_status_1
 md2StatusStruct, [50](#)
zoom_status_2
 md2StatusStruct, [50](#)