

LS-CAT PGPMAC

Generated by Doxygen 1.8.2

Wed Jan 23 2013 10:19:09

Contents

1	The LS-CAT pgpmac Project	1
2	Namespace Index	5
2.1	Namespace List	5
3	Data Structure Index	7
3.1	Data Structures	7
4	File Index	9
4.1	File List	9
5	Namespace Documentation	11
5.1	iniParser Namespace Reference	11
5.1.1	Variable Documentation	11
5.1.1.1	ip	11
5.2	mk_pgpmac_redis Namespace Reference	11
5.2.1	Variable Documentation	12
5.2.1.1	b	12
5.2.1.2	bi_list	12
5.2.1.3	f	12
5.2.1.4	fnc	12
5.2.1.5	hard_ini	12
5.2.1.6	hard_ini_fields	12
5.2.1.7	head	12
5.2.1.8	hi	12
5.2.1.9	i	13
5.2.1.10	motor_dict	13
5.2.1.11	motor_field_lists	13
5.2.1.12	motor_presets	13
5.2.1.13	p	13
5.2.1.14	pi	13
5.2.1.15	ppos	13
5.2.1.16	pref_ini	13

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

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

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

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

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

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

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

6.21.2.38 requested_pos_cnts	52
6.21.2.39 requested_position	52
6.21.2.40 status1	52
6.21.2.41 status1_p	52
6.21.2.42 status2	53
6.21.2.43 status2_p	53
6.21.2.44 status_str	53
6.21.2.45 u2c	53
6.21.2.46 unit	53
6.21.2.47 update_resolution	53
6.21.2.48 win	53
6.21.2.49 write_fmt	53
6.22 Isredis_obj_struct Struct Reference	53
6.22.1 Detailed Description	54
6.22.2 Field Documentation	54
6.22.2.1 avalue	54
6.22.2.2 bvalue	54
6.22.2.3 cond	55
6.22.2.4 cvalue	55
6.22.2.5 dvalue	55
6.22.2.6 events_name	55
6.22.2.7 hits	55
6.22.2.8 key	55
6.22.2.9 lvalue	55
6.22.2.10 mutex	55
6.22.2.11 next	55
6.22.2.12 valid	56
6.22.2.13 value	56
6.22.2.14 value_length	56
6.22.2.15 wait_for_me	56
6.23 Istimer_list_struct Struct Reference	56
6.23.1 Detailed Description	57
6.23.2 Field Documentation	57
6.23.2.1 delay_nsecs	57
6.23.2.2 delay_secs	57
6.23.2.3 event	57
6.23.2.4 init_nsecs	57
6.23.2.5 init_secs	57
6.23.2.6 last_nsecs	57
6.23.2.7 last_secs	57

6.23.2.8	ncalls	58
6.23.2.9	next_nsecs	58
6.23.2.10	next_secs	58
6.23.2.11	shots	58
6.24	md2StatusStruct Struct Reference	58
6.24.1	Detailed Description	59
6.24.2	Field Documentation	60
6.24.2.1	acc11c_1	60
6.24.2.2	acc11c_2	60
6.24.2.3	acc11c_3	60
6.24.2.4	acc11c_5	60
6.24.2.5	acc11c_6	60
6.24.2.6	alignx_act_pos	60
6.24.2.7	alignx_status_1	60
6.24.2.8	alignx_status_2	60
6.24.2.9	aligny_act_pos	60
6.24.2.10	aligny_status_1	60
6.24.2.11	aligny_status_2	60
6.24.2.12	alignz_act_pos	60
6.24.2.13	alignz_status_1	61
6.24.2.14	alignz_status_2	61
6.24.2.15	analyzer_act_pos	61
6.24.2.16	analyzer_status_1	61
6.24.2.17	analyzer_status_2	61
6.24.2.18	aperturey_act_pos	61
6.24.2.19	aperturey_status_1	61
6.24.2.20	aperturey_status_2	61
6.24.2.21	aperturez_act_pos	61
6.24.2.22	aperturez_status_1	61
6.24.2.23	aperturez_status_2	61
6.24.2.24	back_dac	61
6.24.2.25	copy_act_pos	62
6.24.2.26	copy_status_1	62
6.24.2.27	copy_status_2	62
6.24.2.28	capz_act_pos	62
6.24.2.29	capz_status_1	62
6.24.2.30	capz_status_2	62
6.24.2.31	centerx_act_pos	62
6.24.2.32	centerx_status_1	62
6.24.2.33	centerx_status_2	62

6.24.2.34 centery_act_pos	62
6.24.2.35 centery_status_1	62
6.24.2.36 centery_status_2	62
6.24.2.37 dummy1	63
6.24.2.38 dummy2	63
6.24.2.39 dummy3	63
6.24.2.40 dummy4	63
6.24.2.41 dummy5	63
6.24.2.42 dummy6	63
6.24.2.43 dummy7	63
6.24.2.44 dummy8	63
6.24.2.45 dummy9	63
6.24.2.46 dummyA	63
6.24.2.47 dummyB	63
6.24.2.48 front_dac	63
6.24.2.49 fs_has_opened	64
6.24.2.50 fs_has_opened_globally	64
6.24.2.51 fs_is_open	64
6.24.2.52 kappa_act_pos	64
6.24.2.53 kappa_status_1	64
6.24.2.54 kappa_status_2	64
6.24.2.55 moving_flags	64
6.24.2.56 number_passes	64
6.24.2.57 omega_act_pos	64
6.24.2.58 omega_status_1	64
6.24.2.59 omega_status_2	64
6.24.2.60 phi_act_pos	64
6.24.2.61 phi_status_1	65
6.24.2.62 phi_status_2	65
6.24.2.63 phiscan	65
6.24.2.64 scint_act_pos	65
6.24.2.65 scint_piezo	65
6.24.2.66 scint_status_1	65
6.24.2.67 scint_status_2	65
6.24.2.68 zoom_act_pos	65
6.24.2.69 zoom_status_1	65
6.24.2.70 zoom_status_2	65
6.25 tagEthernetCmd Struct Reference	65
6.25.1 Detailed Description	66
6.25.2 Field Documentation	66

6.25.2.1	bData	66
6.25.2.2	Request	66
6.25.2.3	RequestType	66
6.25.2.4	wIndex	66
6.25.2.5	wLength	66
6.25.2.6	wValue	67
7	File Documentation	69
7.1	iniParser.py File Reference	69
7.2	kvredis.c File Reference	69
7.2.1	Macro Definition Documentation	71
7.2.1.1	LS_PG_QUERY_QUEUE_LENGTH	71
7.2.1.2	LS_PG_QUERY_STRING_LENGTH	71
7.2.1.3	LS_PG_STATE_IDLE	71
7.2.1.4	LS_PG_STATE_INIT	71
7.2.1.5	LS_PG_STATE_INIT_POLL	72
7.2.1.6	LS_PG_STATE_RECV	72
7.2.1.7	LS_PG_STATE_RESET	72
7.2.1.8	LS_PG_STATE_RESET_POLL	72
7.2.1.9	LS_PG_STATE_SEND	72
7.2.1.10	LS_PG_STATE_SEND_FLUSH	72
7.2.2	Typedef Documentation	72
7.2.2.1	lspg_query_queue_t	72
7.2.3	Function Documentation	72
7.2.3.1	addRead	72
7.2.3.2	addWrite	72
7.2.3.3	cleanup	73
7.2.3.4	debugCB	73
7.2.3.5	delRead	73
7.2.3.6	delWrite	74
7.2.3.7	fd_service	74
7.2.3.8	lspg_allkvs_cb	74
7.2.3.9	lspg_flush	75
7.2.3.10	lspg_next_state	75
7.2.3.11	lspg_notice_processor	76
7.2.3.12	lspg_pg_connect	76
7.2.3.13	lspg_pg_service	77
7.2.3.14	lspg_query_next	79
7.2.3.15	lspg_query_push	79
7.2.3.16	lspg_query_reply_next	79

7.2.3.17	lspg_query_reply_peek	80
7.2.3.18	lspg_receive	80
7.2.3.19	lspg_send_next_query	81
7.2.3.20	main	81
7.2.3.21	redisDisconnectCB	83
7.2.4	Variable Documentation	83
7.2.4.1	cmdac	83
7.2.4.2	cmdfd	83
7.2.4.3	kvseq	83
7.2.4.4	ls_pg_state	83
7.2.4.5	lspg_connectPoll_response	84
7.2.4.6	lspg_query_queue	84
7.2.4.7	lspg_query_queue_off	84
7.2.4.8	lspg_query_queue_on	84
7.2.4.9	lspg_query_queue_reply	84
7.2.4.10	lspg_resetPoll_response	84
7.2.4.11	lspgfd	84
7.2.4.12	now	84
7.2.4.13	q	85
7.2.4.14	subac	85
7.2.4.15	subfd	85
7.3	lsevents.c File Reference	85
7.3.1	Detailed Description	86
7.3.2	Macro Definition Documentation	86
7.3.2.1	LSEVENTS_QUEUE_LENGTH	86
7.3.3	Typedef Documentation	87
7.3.3.1	lsevents_listener_t	87
7.3.3.2	lsevents_queue_t	87
7.3.4	Function Documentation	87
7.3.4.1	lsevents_add_listener	87
7.3.4.2	lsevents_init	88
7.3.4.3	lsevents_remove_listener	88
7.3.4.4	lsevents_run	88
7.3.4.5	lsevents_send_event	89
7.3.4.6	lsevents_worker	89
7.3.5	Variable Documentation	90
7.3.5.1	lsevents_listener_mutex	90
7.3.5.2	lsevents_listeners_p	90
7.3.5.3	lsevents_queue	90
7.3.5.4	lsevents_queue_cond	90

7.3.5.5	lsevents_queue_mutex	90
7.3.5.6	lsevents_queue_off	91
7.3.5.7	lsevents_queue_on	91
7.3.5.8	lsevents_thread	91
7.4	lslogging.c File Reference	91
7.4.1	Detailed Description	92
7.4.2	Macro Definition Documentation	92
7.4.2.1	LSLOGGING_FILE_NAME	92
7.4.2.2	LSLOGGING_MSG_LENGTH	92
7.4.2.3	LSLOGGING_QUEUE_LENGTH	93
7.4.3	Typedef Documentation	93
7.4.3.1	lslogging_queue_t	93
7.4.4	Function Documentation	93
7.4.4.1	lslogging_init	93
7.4.4.2	lslogging_log_message	93
7.4.4.3	lslogging_run	94
7.4.4.4	lslogging_worker	94
7.4.5	Variable Documentation	94
7.4.5.1	lslogging_cond	94
7.4.5.2	lslogging_file	94
7.4.5.3	lslogging_mutex	95
7.4.5.4	lslogging_off	95
7.4.5.5	lslogging_on	95
7.4.5.6	lslogging_queue	95
7.4.5.7	lslogging_thread	95
7.5	lspg.c File Reference	95
7.5.1	Detailed Description	100
7.5.2	Macro Definition Documentation	100
7.5.2.1	LS_PG_QUERY_QUEUE_LENGTH	100
7.5.2.2	LS_PG_STATE_IDLE	101
7.5.2.3	LS_PG_STATE_INIT	101
7.5.2.4	LS_PG_STATE_INIT_POLL	101
7.5.2.5	LS_PG_STATE_RECV	101
7.5.2.6	LS_PG_STATE_RESET	101
7.5.2.7	LS_PG_STATE_RESET_POLL	101
7.5.2.8	LS_PG_STATE_SEND	101
7.5.2.9	LS_PG_STATE_SEND_FLUSH	101
7.5.3	Typedef Documentation	101
7.5.3.1	lspg_lock_detector_t	101
7.5.3.2	lspg_lock_diffractionmeter_t	101

7.5.3.3	lspg_seq_run_prep_t	101
7.5.3.4	lspg_wait_for_detector_t	101
7.5.4	Function Documentation	102
7.5.4.1	lspg_array2ptrs	102
7.5.4.2	lspg_cmd_cb	103
7.5.4.3	lspg_demandairrights_all	104
7.5.4.4	lspg_demandairrights_call	104
7.5.4.5	lspg_demandairrights_cb	104
7.5.4.6	lspg_demandairrights_init	104
7.5.4.7	lspg_demandairrights_wait	104
7.5.4.8	lspg_flush	105
7.5.4.9	lspg_getcenter_all	105
7.5.4.10	lspg_getcenter_call	105
7.5.4.11	lspg_getcenter_cb	105
7.5.4.12	lspg_getcenter_done	106
7.5.4.13	lspg_getcenter_init	106
7.5.4.14	lspg_getcenter_wait	107
7.5.4.15	lspg_getcurrentsampleid_call	107
7.5.4.16	lspg_getcurrentsampleid_cb	107
7.5.4.17	lspg_getcurrentsampleid_init	108
7.5.4.18	lspg_getcurrentsampleid_read	108
7.5.4.19	lspg_getcurrentsampleid_wait_for_id	108
7.5.4.20	lspg_init	108
7.5.4.21	lspg_lock_detector_all	109
7.5.4.22	lspg_lock_detector_call	109
7.5.4.23	lspg_lock_detector_cb	109
7.5.4.24	lspg_lock_detector_done	109
7.5.4.25	lspg_lock_detector_init	110
7.5.4.26	lspg_lock_detector_wait	110
7.5.4.27	lspg_lock_diffractionmeter_all	110
7.5.4.28	lspg_lock_diffractionmeter_call	110
7.5.4.29	lspg_lock_diffractionmeter_cb	110
7.5.4.30	lspg_lock_diffractionmeter_done	111
7.5.4.31	lspg_lock_diffractionmeter_init	111
7.5.4.32	lspg_lock_diffractionmeter_wait	111
7.5.4.33	lspg_next_state	111
7.5.4.34	lspg_nextaction_cb	112
7.5.4.35	lspg_nextsample_all	113
7.5.4.36	lspg_nextsample_call	113
7.5.4.37	lspg_nextsample_cb	113

7.5.4.38	lspg_nextsample_done	114
7.5.4.39	lspg_nextsample_init	114
7.5.4.40	lspg_nextsample_wait	114
7.5.4.41	lspg_nextshot_call	114
7.5.4.42	lspg_nextshot_cb	115
7.5.4.43	lspg_nextshot_done	118
7.5.4.44	lspg_nextshot_init	119
7.5.4.45	lspg_nextshot_wait	119
7.5.4.46	lspg_notice_processor	119
7.5.4.47	lspg_pg_connect	119
7.5.4.48	lspg_pg_service	120
7.5.4.49	lspg_query_next	122
7.5.4.50	lspg_query_push	122
7.5.4.51	lspg_query_reply_next	123
7.5.4.52	lspg_query_reply_peek	123
7.5.4.53	lspg_receive	123
7.5.4.54	lspg_run	124
7.5.4.55	lspg_send_next_query	124
7.5.4.56	lspg_seq_run_prep_all	125
7.5.4.57	lspg_seq_run_prep_call	126
7.5.4.58	lspg_seq_run_prep_cb	126
7.5.4.59	lspg_seq_run_prep_done	126
7.5.4.60	lspg_seq_run_prep_init	126
7.5.4.61	lspg_seq_run_prep_wait	127
7.5.4.62	lspg_sig_service	127
7.5.4.63	lspg_starttransfer_all	127
7.5.4.64	lspg_starttransfer_call	128
7.5.4.65	lspg_starttransfer_cb	128
7.5.4.66	lspg_starttransfer_done	128
7.5.4.67	lspg_starttransfer_init	128
7.5.4.68	lspg_starttransfer_wait	129
7.5.4.69	lspg_wait_for_detector_all	129
7.5.4.70	lspg_wait_for_detector_call	129
7.5.4.71	lspg_wait_for_detector_cb	129
7.5.4.72	lspg_wait_for_detector_done	129
7.5.4.73	lspg_wait_for_detector_init	130
7.5.4.74	lspg_wait_for_detector_wait	130
7.5.4.75	lspg_waitcryo_all	130
7.5.4.76	lspg_waitcryo_cb	130
7.5.4.77	lspg_waitcryo_init	131

7.5.4.78	lspg_worker	131
7.5.4.79	lspmac_sample_detector_cb	132
7.5.5	Variable Documentation	132
7.5.5.1	ls_pg_state	132
7.5.5.2	lspg_connectPoll_response	132
7.5.5.3	lspg_demandairrights	132
7.5.5.4	lspg_getcenter	132
7.5.5.5	lspg_getcurrentsampleid	132
7.5.5.6	lspg_lock_detector	133
7.5.5.7	lspg_lock_diffractionmeter	133
7.5.5.8	lspg_nextsample	133
7.5.5.9	lspg_nextshot	133
7.5.5.10	lspg_query_queue	133
7.5.5.11	lspg_query_queue_off	133
7.5.5.12	lspg_query_queue_on	133
7.5.5.13	lspg_query_queue_reply	133
7.5.5.14	lspg_queue_cond	133
7.5.5.15	lspg_queue_mutex	134
7.5.5.16	lspg_resetPoll_response	134
7.5.5.17	lspg_seq_run_prep	134
7.5.5.18	lspg_starttransfer	134
7.5.5.19	lspg_thread	134
7.5.5.20	lspg_wait_for_detector	134
7.5.5.21	lspg_waitcryo	134
7.5.5.22	lspgfd	134
7.5.5.23	now	134
7.5.5.24	q	135
7.6	lspmac.c File Reference	135
7.6.1	Detailed Description	143
7.6.2	Macro Definition Documentation	143
7.6.2.1	LS_PMAC_STATE_CR	143
7.6.2.2	LS_PMAC_STATE_DETACHED	144
7.6.2.3	LS_PMAC_STATE_GB	144
7.6.2.4	LS_PMAC_STATE_GMR	144
7.6.2.5	LS_PMAC_STATE_IDLE	144
7.6.2.6	LS_PMAC_STATE_RESET	144
7.6.2.7	LS_PMAC_STATE_RR	144
7.6.2.8	LS_PMAC_STATE_SC	144
7.6.2.9	LS_PMAC_STATE_WACK	144
7.6.2.10	LS_PMAC_STATE_WACK_CC	144

7.6.2.11	LS_PMAC_STATE_WACK_NFR	144
7.6.2.12	LS_PMAC_STATE_WACK_RR	144
7.6.2.13	LS_PMAC_STATE_WCR	144
7.6.2.14	LS_PMAC_STATE_WGB	145
7.6.2.15	LSPMAC_DPASCII_QUEUE_LENGTH	145
7.6.2.16	LSPMAC_PRESET_REGEX	145
7.6.2.17	PMAC_CMD_QUEUE_LENGTH	145
7.6.2.18	pmac_cmd_size	145
7.6.2.19	PMAC_MIN_CMD_TIME	145
7.6.2.20	PMACPORT	145
7.6.2.21	VR_CTRL_RESPONSE	145
7.6.2.22	VR_DOWNLOAD	145
7.6.2.23	VR_FWDOWNLOAD	145
7.6.2.24	VR_IPADDRESS	145
7.6.2.25	VR_PMAC_FLUSH	146
7.6.2.26	VR_PMAC_GETBUFFER	146
7.6.2.27	VR_PMAC_GETLINE	146
7.6.2.28	VR_PMAC_GETMEM	146
7.6.2.29	VR_PMAC_GETRESPONSE	146
7.6.2.30	VR_PMAC_PORT	146
7.6.2.31	VR_PMAC_READREADY	146
7.6.2.32	VR_PMAC_SENDCTRLCHAR	146
7.6.2.33	VR_PMAC_SENDLINE	146
7.6.2.34	VR_PMAC_SETBIT	146
7.6.2.35	VR_PMAC_SETBITS	146
7.6.2.36	VR_PMAC_SETMEM	146
7.6.2.37	VR_PMAC_WRITEBUFFER	147
7.6.2.38	VR_PMAC_WRITEERROR	147
7.6.2.39	VR_UPLOAD	147
7.6.3	Typedef Documentation	147
7.6.3.1	lspmac_ascii_buffers_t	147
7.6.3.2	lspmac_dpascii_queue_t	147
7.6.3.3	md2_status_t	147
7.6.4	Function Documentation	147
7.6.4.1	_lspmac_motor_init	147
7.6.4.2	cleanstr	148
7.6.4.3	hex_dump	148
7.6.4.4	lsConnect	149
7.6.4.5	lspmac_abort	150
7.6.4.6	lspmac_asciicmdCB	150

7.6.4.7	lspmac_backLight_down_cb	150
7.6.4.8	lspmac_backLight_up_cb	150
7.6.4.9	lspmac_bi_init	151
7.6.4.10	lspmac_blight_lut_setup	151
7.6.4.11	lspmac_bo_init	151
7.6.4.12	lspmac_bo_read	152
7.6.4.13	lspmac_command_done_cb	152
7.6.4.14	lspmac_cryoSwitchChanged_cb	153
7.6.4.15	lspmac_dac_init	153
7.6.4.16	lspmac_dac_read	153
7.6.4.17	lspmac_Error	154
7.6.4.18	lspmac_est_move_time	154
7.6.4.19	lspmac_flight_lut_setup	155
7.6.4.20	lspmac_fscint_lut_setup	156
7.6.4.21	lspmac_fshut_init	156
7.6.4.22	lspmac_get_ascii	157
7.6.4.23	lspmac_get_ascii_cb	157
7.6.4.24	lspmac_get_status	158
7.6.4.25	lspmac_get_status_cb	159
7.6.4.26	lspmac_GetAllIVars	162
7.6.4.27	lspmac_GetAllIVarsCB	162
7.6.4.28	lspmac_GetAllMVars	162
7.6.4.29	lspmac_GetAllMVarsCB	162
7.6.4.30	lspmac_getBIPosition	163
7.6.4.31	lspmac_Getmem	163
7.6.4.32	lspmac_GetmemReplyCB	163
7.6.4.33	lspmac_getPosition	163
7.6.4.34	lspmac_GetShortReplyCB	164
7.6.4.35	lspmac_home1_queue	164
7.6.4.36	lspmac_home2_queue	166
7.6.4.37	lspmac_init	166
7.6.4.38	lspmac_jogabs_queue	169
7.6.4.39	lspmac_light_zoom_cb	170
7.6.4.40	lspmac_lut	170
7.6.4.41	lspmac_more_ascii_cb	171
7.6.4.42	lspmac_motor_init	171
7.6.4.43	lspmac_move_or_jog_abs_queue	172
7.6.4.44	lspmac_move_or_jog_preset_queue	174
7.6.4.45	lspmac_move_preset_queue	175
7.6.4.46	lspmac_moveabs_blight_factor_queue	175

7.6.4.47	lspmac_moveabs_bo_queue	176
7.6.4.48	lspmac_moveabs_flight_factor_queue	176
7.6.4.49	lspmac_moveabs_frontlight_oo_queue	176
7.6.4.50	lspmac_moveabs_fshut_queue	177
7.6.4.51	lspmac_moveabs_queue	177
7.6.4.52	lspmac_moveabs_timed_queue	178
7.6.4.53	lspmac_moveabs_wait	178
7.6.4.54	lspmac_movedac_queue	180
7.6.4.55	lspmac_movezoom_queue	180
7.6.4.56	lspmac_next_state	181
7.6.4.57	lspmac_pmacmotor_read	182
7.6.4.58	lspmac_pop_queue	186
7.6.4.59	lspmac_pop_reply	186
7.6.4.60	lspmac_push_queue	186
7.6.4.61	lspmac_Reset	187
7.6.4.62	lspmac_reset_queue	187
7.6.4.63	lspmac_rlut	187
7.6.4.64	lspmac_run	188
7.6.4.65	lspmac_scint_dried_cb	189
7.6.4.66	lspmac_scint_inPosition_cb	190
7.6.4.67	lspmac_send_command	190
7.6.4.68	lspmac_sendcmd	191
7.6.4.69	lspmac_sendcmd_nocb	191
7.6.4.70	lspmac_SendControlReplyPrintCB	192
7.6.4.71	lspmac_Service	192
7.6.4.72	lspmac_shutter_read	195
7.6.4.73	lspmac_SockFlush	195
7.6.4.74	lspmac_SockGetmem	195
7.6.4.75	lspmac_SockSendControlCharPrint	196
7.6.4.76	lspmac_SockSendDPControlChar	196
7.6.4.77	lspmac_SockSendDPControlCharCB	196
7.6.4.78	lspmac_SockSendDPLine	196
7.6.4.79	lspmac_SockSendDPQueue	197
7.6.4.80	lspmac_SockSendline	197
7.6.4.81	lspmac_SockSendline_nr	198
7.6.4.82	lspmac_soft_motor_init	198
7.6.4.83	lspmac_soft_motor_read	198
7.6.4.84	lspmac_test_preset	199
7.6.4.85	lspmac_video_rotate	199
7.6.4.86	lspmac_worker	199

7.6.4.87	lspmac_zoom_lut_setup	200
7.6.5	Variable Documentation	201
7.6.5.1	alignx	201
7.6.5.2	aligny	201
7.6.5.3	alignz	201
7.6.5.4	anal	201
7.6.5.5	apery	201
7.6.5.6	aperz	201
7.6.5.7	arm_parked	201
7.6.5.8	blight	201
7.6.5.9	blight_down	201
7.6.5.10	blight_f	202
7.6.5.11	blight_ud	202
7.6.5.12	blight_up	202
7.6.5.13	capy	202
7.6.5.14	capz	202
7.6.5.15	cenx	202
7.6.5.16	ceny	202
7.6.5.17	cr_cmd	202
7.6.5.18	cryo	202
7.6.5.19	cryo_back	203
7.6.5.20	cryo_switch	203
7.6.5.21	dbmem	203
7.6.5.22	dbmemIn	203
7.6.5.23	dryer	203
7.6.5.24	etel_init_ok	203
7.6.5.25	etel_on	203
7.6.5.26	etel_ready	203
7.6.5.27	ethCmdOff	203
7.6.5.28	ethCmdOn	204
7.6.5.29	ethCmdQueue	204
7.6.5.30	ethCmdReply	204
7.6.5.31	flight	204
7.6.5.32	flight_f	204
7.6.5.33	flight_oo	204
7.6.5.34	fluo	204
7.6.5.35	fluor_back	204
7.6.5.36	fscint	204
7.6.5.37	fshut	205
7.6.5.38	gb_cmd	205

7.6.5.39	getivars	205
7.6.5.40	getmvars	205
7.6.5.41	hp_air	205
7.6.5.42	kappa	205
7.6.5.43	lp_air	205
7.6.5.44	ls_pmac_state	205
7.6.5.45	lspmac_ascii_buffers	205
7.6.5.46	lspmac_ascii_buffers_mutex	205
7.6.5.47	lspmac_ascii_busy	206
7.6.5.48	lspmac_ascii_mutex	206
7.6.5.49	lspmac_bis	206
7.6.5.50	lspmac_dpascii_off	206
7.6.5.51	lspmac_dpascii_on	206
7.6.5.52	lspmac_dpascii_queue	206
7.6.5.53	lspmac_motors	206
7.6.5.54	lspmac_moving_cond	206
7.6.5.55	lspmac_moving_flags	206
7.6.5.56	lspmac_moving_mutex	206
7.6.5.57	lspmac_nbis	207
7.6.5.58	lspmac_nmotors	207
7.6.5.59	lspmac_shutter_cond	207
7.6.5.60	lspmac_shutter_has_opened	207
7.6.5.61	lspmac_shutter_mutex	207
7.6.5.62	lspmac_shutter_state	207
7.6.5.63	lspmac_status_last_time	207
7.6.5.64	lspmac_status_time	207
7.6.5.65	md2_status	207
7.6.5.66	md2_status_mutex	208
7.6.5.67	minikappa_ok	208
7.6.5.68	now	208
7.6.5.69	omega	208
7.6.5.70	omega_zero_search	208
7.6.5.71	omega_zero_time	208
7.6.5.72	omega_zero_velocity	208
7.6.5.73	phi	208
7.6.5.74	pmac_error_strs	208
7.6.5.75	pmac_queue_cond	209
7.6.5.76	pmac_queue_mutex	209
7.6.5.77	pmac_thread	209
7.6.5.78	pmacfd	209

7.6.5.79	rr_cmd	209
7.6.5.80	sample_detected	209
7.6.5.81	scint	209
7.6.5.82	shutter_open	210
7.6.5.83	smart_mag_err	210
7.6.5.84	smart_mag_off	210
7.6.5.85	smart_mag_on	210
7.6.5.86	smart_mag_oo	210
7.6.5.87	zoom	210
7.7	Isredis.c File Reference	210
7.7.1	Detailed Description	212
7.7.2	Function Documentation	213
7.7.2.1	_Isredis_get_obj	213
7.7.2.2	_Isredis_set_value	214
7.7.2.3	Isredis_addRead	215
7.7.2.4	Isredis_addWrite	215
7.7.2.5	Isredis_cleanup	215
7.7.2.6	Isredis_cmpnstr	216
7.7.2.7	Isredis_cmpstr	216
7.7.2.8	Isredis_debugCB	216
7.7.2.9	Isredis_delRead	217
7.7.2.10	Isredis_delWrite	217
7.7.2.11	Isredis_fd_service	217
7.7.2.12	Isredis_find_preset	218
7.7.2.13	Isredis_get_obj	219
7.7.2.14	Isredis_get_string_array	219
7.7.2.15	Isredis_getb	219
7.7.2.16	Isredis_getc	220
7.7.2.17	Isredis_getd	220
7.7.2.18	Isredis_getl	220
7.7.2.19	Isredis_getstr	220
7.7.2.20	Isredis_hgetCB	221
7.7.2.21	Isredis_init	221
7.7.2.22	Isredis_keysCB	222
7.7.2.23	Isredis_maybe_add_key	223
7.7.2.24	Isredis_regexec	223
7.7.2.25	Isredis_run	223
7.7.2.26	Isredis_set_value	223
7.7.2.27	Isredis_setstr	224
7.7.2.28	Isredis_sig_service	224

7.7.2.29	lsredis_subCB	225
7.7.2.30	lsredis_worker	226
7.7.2.31	redisDisconnectCB	227
7.7.3	Variable Documentation	227
7.7.3.1	lsredis_cond	227
7.7.3.2	lsredis_head	227
7.7.3.3	lsredis_htab	228
7.7.3.4	lsredis_key_select_regex	228
7.7.3.5	lsredis_mutex	228
7.7.3.6	lsredis_objs	228
7.7.3.7	lsredis_publisher	228
7.7.3.8	lsredis_running	228
7.7.3.9	lsredis_thread	228
7.7.3.10	roac	228
7.7.3.11	rofd	228
7.7.3.12	subac	228
7.7.3.13	subfd	228
7.7.3.14	wrac	228
7.7.3.15	wrfd	229
7.8	lstimer.c File Reference	229
7.8.1	Detailed Description	230
7.8.2	Macro Definition Documentation	230
7.8.2.1	LSTIMER_LIST_LENGTH	230
7.8.2.2	LSTIMER_RESOLUTION_NSECS	230
7.8.3	Typedef Documentation	230
7.8.3.1	lstimer_list_t	230
7.8.4	Function Documentation	230
7.8.4.1	handler	230
7.8.4.2	lstimer_add_timer	231
7.8.4.3	lstimer_init	232
7.8.4.4	lstimer_run	232
7.8.4.5	lstimer_worker	232
7.8.4.6	service_timers	233
7.8.5	Variable Documentation	234
7.8.5.1	lstimer_active_timers	234
7.8.5.2	lstimer_cond	234
7.8.5.3	lstimer_list	234
7.8.5.4	lstimer_mutex	234
7.8.5.5	lstimer_thread	235
7.8.5.6	lstimer_timerid	235

7.8.5.7	new_timer	235
7.9	md2cmds.c File Reference	235
7.9.1	Detailed Description	237
7.9.2	Function Documentation	237
7.9.2.1	md2cmds_abort	237
7.9.2.2	md2cmds_action_queue	237
7.9.2.3	md2cmds_action_wait	238
7.9.2.4	md2cmds_center	238
7.9.2.5	md2cmds_collect	238
7.9.2.6	md2cmds_coordsys_1_stopped_cb	242
7.9.2.7	md2cmds_coordsys_2_stopped_cb	242
7.9.2.8	md2cmds_coordsys_3_stopped_cb	242
7.9.2.9	md2cmds_coordsys_4_stopped_cb	242
7.9.2.10	md2cmds_coordsys_5_stopped_cb	243
7.9.2.11	md2cmds_coordsys_7_stopped_cb	243
7.9.2.12	md2cmds_home_prep	243
7.9.2.13	md2cmds_home_wait	243
7.9.2.14	md2cmds_init	244
7.9.2.15	md2cmds_is_moving	244
7.9.2.16	md2cmds_kappaphi_move	244
7.9.2.17	md2cmds_maybe_done_homing_cb	245
7.9.2.18	md2cmds_maybe_done_moving_cb	245
7.9.2.19	md2cmds_maybe_rotate_done_cb	245
7.9.2.20	md2cmds_move_prep	246
7.9.2.21	md2cmds_move_wait	246
7.9.2.22	md2cmds_moveAbs	246
7.9.2.23	md2cmds_mvcenter_move	248
7.9.2.24	md2cmds_organs_move_presets	248
7.9.2.25	md2cmds_phase_change	249
7.9.2.26	md2cmds_prep_axis	251
7.9.2.27	md2cmds_rotate	252
7.9.2.28	md2cmds_rotate_cb	254
7.9.2.29	md2cmds_run	254
7.9.2.30	md2cmds_set_scale_cb	254
7.9.2.31	md2cmds_time_capz_cb	255
7.9.2.32	md2cmds_transfer	255
7.9.2.33	md2cmds_worker	258
7.9.3	Variable Documentation	258
7.9.3.1	md2cmds_capz_moving_time	258
7.9.3.2	md2cmds_cmd	258

7.9.3.3	md2cmds_cond	258
7.9.3.4	md2cmds_homing_cond	259
7.9.3.5	md2cmds_homing_count	259
7.9.3.6	md2cmds_homing_mutex	259
7.9.3.7	md2cmds_md_status_code	259
7.9.3.8	md2cmds_moving_cond	259
7.9.3.9	md2cmds_moving_count	259
7.9.3.10	md2cmds_moving_mutex	259
7.9.3.11	md2cmds_moving_queue_wait	259
7.9.3.12	md2cmds_mutex	259
7.9.3.13	md2cmds_thread	259
7.9.3.14	rotating	260
7.10	mk_pgpmac_redis.py File Reference	260
7.11	pgpmac.c File Reference	260
7.11.1	Detailed Description	261
7.11.2	Function Documentation	261
7.11.2.1	main	261
7.11.2.2	pgpmac_printf	263
7.11.2.3	stdinService	263
7.11.3	Variable Documentation	264
7.11.3.1	ncurses_mutex	264
7.11.3.2	running	265
7.11.3.3	stdinfda	265
7.11.3.4	term_input	265
7.11.3.5	term_output	265
7.11.3.6	term_status	265
7.11.3.7	term_status2	265
7.12	pgpmac.h File Reference	265
7.12.1	Detailed Description	273
7.12.2	Macro Definition Documentation	273
7.12.2.1	_GNU_SOURCE	273
7.12.2.2	LS_DISPLAY_WINDOW_HEIGHT	273
7.12.2.3	LS_DISPLAY_WINDOW_WIDTH	273
7.12.2.4	LS_PG_QUERY_STRING_LENGTH	273
7.12.2.5	LSEVENTS_EVENT_LENGTH	273
7.12.2.6	MD2CMDS_CMD_LENGTH	274
7.12.3	Typedef Documentation	274
7.12.3.1	lspg_demandairrights_t	274
7.12.3.2	lspg_getcenter_t	274
7.12.3.3	lspg_getcurrentsampleid_t	274

7.12.3.4	lspg_nextsample_t	274
7.12.3.5	lspg_nextshot_t	274
7.12.3.6	lspg_query_queue_t	274
7.12.3.7	lspg_starttransfer_t	274
7.12.3.8	lspg_waitcryo_t	274
7.12.3.9	lspmac_bi_t	274
7.12.3.10	lspmac_motor_t	274
7.12.3.11	lsredis_obj_t	274
7.12.3.12	pmac_cmd_queue_t	275
7.12.3.13	pmac_cmd_t	275
7.12.4	Function Documentation	275
7.12.4.1	lsevents_add_listener	275
7.12.4.2	lsevents_init	276
7.12.4.3	lsevents_remove_listener	276
7.12.4.4	lsevents_run	276
7.12.4.5	lsevents_send_event	277
7.12.4.6	lslogging_init	277
7.12.4.7	lslogging_log_message	277
7.12.4.8	lslogging_run	278
7.12.4.9	lspg_array2ptrs	278
7.12.4.10	lspg_demandairrights_all	280
7.12.4.11	lspg_getcenter_call	280
7.12.4.12	lspg_getcenter_done	280
7.12.4.13	lspg_getcenter_wait	280
7.12.4.14	lspg_getcurrentsampleid_wait_for_id	280
7.12.4.15	lspg_init	281
7.12.4.16	lspg_nextsample_all	281
7.12.4.17	lspg_nextshot_call	281
7.12.4.18	lspg_nextshot_done	281
7.12.4.19	lspg_nextshot_wait	282
7.12.4.20	lspg_query_push	282
7.12.4.21	lspg_run	282
7.12.4.22	lspg_seq_run_prep_all	283
7.12.4.23	lspg_starttransfer_call	283
7.12.4.24	lspg_starttransfer_done	283
7.12.4.25	lspg_starttransfer_wait	283
7.12.4.26	lspg_waitcryo_all	284
7.12.4.27	lspg_waitcryo_cb	284
7.12.4.28	lspg_zoom_lut_call	284
7.12.4.29	lspmac_abort	284

7.12.4.30	lspmac_getBIPosition	284
7.12.4.31	lspmac_getPosition	285
7.12.4.32	lspmac_home1_queue	285
7.12.4.33	lspmac_init	286
7.12.4.34	lspmac_jogabs_queue	289
7.12.4.35	lspmac_move_or_jog_abs_queue	289
7.12.4.36	lspmac_move_or_jog_preset_queue	292
7.12.4.37	lspmac_move_or_jog_queue	292
7.12.4.38	lspmac_move_preset_queue	292
7.12.4.39	lspmac_moveabs_queue	293
7.12.4.40	lspmac_moveabs_wait	293
7.12.4.41	lspmac_run	294
7.12.4.42	lspmac_SockSendDPLine	295
7.12.4.43	lspmac_SockSendline	296
7.12.4.44	lspmac_video_rotate	296
7.12.4.45	lsredis_cmpnstr	297
7.12.4.46	lsredis_cmpstr	297
7.12.4.47	lsredis_find_preset	297
7.12.4.48	lsredis_get_obj	298
7.12.4.49	lsredis_get_string_array	298
7.12.4.50	lsredis_getb	299
7.12.4.51	lsredis_getd	299
7.12.4.52	lsredis_getl	299
7.12.4.53	lsredis_getstr	299
7.12.4.54	lsredis_init	300
7.12.4.55	lsredis_regexec	301
7.12.4.56	lsredis_run	301
7.12.4.57	lsredis_setstr	301
7.12.4.58	lster_timer_add_timer	302
7.12.4.59	lster_timer_init	303
7.12.4.60	lster_timer_run	303
7.12.4.61	lsupdate_init	304
7.12.4.62	lsupdate_run	304
7.12.4.63	md2cmds_init	304
7.12.4.64	md2cmds_run	304
7.12.4.65	pgpmac_printf	304
7.12.4.66	PmacSockSendline	305
7.12.5	Variable Documentation	305
7.12.5.1	alignx	305
7.12.5.2	aligny	305

7.12.5.3 alignz	305
7.12.5.4 anal	305
7.12.5.5 apery	305
7.12.5.6 aperz	305
7.12.5.7 arm_parked	306
7.12.5.8 blight	306
7.12.5.9 blight_down	306
7.12.5.10 blight_f	306
7.12.5.11 blight_ud	306
7.12.5.12 blight_up	306
7.12.5.13 capy	306
7.12.5.14 capz	306
7.12.5.15 cenx	306
7.12.5.16 ceny	307
7.12.5.17 cryo	307
7.12.5.18 cryo_back	307
7.12.5.19 cryo_switch	307
7.12.5.20 dryer	307
7.12.5.21 etel_init_ok	307
7.12.5.22 etel_on	307
7.12.5.23 etel_ready	307
7.12.5.24 flight	307
7.12.5.25 flight_f	308
7.12.5.26 flight_oo	308
7.12.5.27 fluo	308
7.12.5.28 fluor_back	308
7.12.5.29 fscint	308
7.12.5.30 fshut	308
7.12.5.31 hp_air	308
7.12.5.32 kappa	308
7.12.5.33 lp_air	308
7.12.5.34 lspg_demandairrights	309
7.12.5.35 lspg_getcenter	309
7.12.5.36 lspg_getcurrentsampleid	309
7.12.5.37 lspg_nextsample	309
7.12.5.38 lspg_nextshot	309
7.12.5.39 lspg_starttransfer	309
7.12.5.40 lspg_waitcryo	309
7.12.5.41 lspmac_motors	309
7.12.5.42 lspmac_moving_cond	309

7.12.5.43 lspmact_moving_flags	310
7.12.5.44 lspmact_moving_mutex	310
7.12.5.45 lspmact_nmotors	310
7.12.5.46 lspmact_shutter_cond	310
7.12.5.47 lspmact_shutter_has_opened	310
7.12.5.48 lspmact_shutter_mutex	310
7.12.5.49 lspmact_shutter_state	310
7.12.5.50 md2_status_mutex	310
7.12.5.51 md2cmds_cmd	310
7.12.5.52 md2cmds_cond	311
7.12.5.53 md2cmds_md_status_code	311
7.12.5.54 md2cmds_mutex	311
7.12.5.55 md2cmds_pg_cond	311
7.12.5.56 md2cmds_pg_mutex	311
7.12.5.57 minikappa_ok	311
7.12.5.58 ncurses_mutex	311
7.12.5.59 omega	311
7.12.5.60 omega_zero_time	311
7.12.5.61 phi	311
7.12.5.62 pmact_queue_cond	311
7.12.5.63 pmact_queue_mutex	312
7.12.5.64 sample_detected	312
7.12.5.65 scint	312
7.12.5.66 shutter_open	312
7.12.5.67 smart_mag_err	312
7.12.5.68 smart_mag_off	312
7.12.5.69 smart_mag_on	312
7.12.5.70 smart_mag_oo	312
7.12.5.71 term_input	312
7.12.5.72 term_output	313
7.12.5.73 term_status	313
7.12.5.74 term_status2	313
7.12.5.75 zoom	313

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
lsredis.c	Receive key value pair updates from redis databases
lslogging.c	A logging utility to simplify debugging
lspg.c	Handles communications with the controlling 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 exectutive 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

Namespace Index

2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

iniParser	11
mk_pgpmac_redis	11

Chapter 3

Data Structure Index

3.1 Data Structures

Here are the data structures with brief descriptions:

iniParser.iniParser	This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version	15
lsevents_listener_struct	Linked list of event listeners	17
lsevents_queue_struct	Storage definition for the events	18
lslogging_queue_struct	Our log object: time and message	19
lspg_demandairrights_struct	20
lspg_getcenter_struct	Storage for getcenter query Used for the md2 ROTATE command that generates the centering movies	20
lspg_getcurrentsampleid_struct	23
lspg_lock_detector_struct	Lock detector object Implements detector lock for exposure control	24
lspg_lock_diffractionmeter_struct	Object used to impliment locking the diffractometer Critical to exposure timing	25
lspg_nextsample_struct	Returns the next sample number Just a 32 bit int (Ha!, take that, nextshot!)	25
lspg_nextshot_struct	Storage definition for nextshot query	27
lspg_seq_run_prep_struct	Data collection running object	37
lspg_starttransfer_struct	Returns 1 if transfer can continue 0 to abort	38
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	39
lspg_waitcryo_struct	40
lspgQueryQueueStruct	Store each query along with it's callback function	41
lspmac_asci_buffers_struct	42
lspmac_bi_struct	Storage for binary inputs	42
lspmac_cmd_queue_struct	PMAC command queue item	44

lspmac_dpascii_queue_struct	45
lspmac_motor_struct	
Motor information	46
lsredis_obj_struct	
Redis Object Basic object whose value is synchronized with our redis db	53
lstimer_list_struct	
Everything we need to know about a timer	56
md2StatusStruct	
The block of memory retrieved in a status request	58
tagEthernetCmd	
PMAC ethernet packet definition	65

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

iniParser.py	69
kvredis.c	69
lsevents.c	
Event subsystem for inter-pgpmac communication	85
lslogging.c	
Logs messages to a file	91
lspg.c	
Postgresql support for the LS-CAT pgpmac project	95
lspmac.c	
Routines concerned with communication with PMAC	135
lsredis.c	
Support redis hash synchronization	210
lstimer.c	
Support for delayed and periodic events	229
md2cmds.c	
Implements commands to run the md2 diffractometer attached to a PMAC controled by post- gresql	235
mk_pgpmac_redis.py	260
pgpmac.c	
Main for the pgpmac project	260
pgpmac.h	
Headers for the entire pgpmac project	265

Chapter 5

Namespace Documentation

5.1 iniParser Namespace Reference

Data Structures

- class [iniParser](#)

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

Variables

- tuple [ip](#) [iniParser](#)("21-ID-E/microdiff_hard.ini")

5.1.1 Variable Documentation

5.1.1.1 tuple [iniParser.ip](#) [iniParser](#)("21-ID-E/microdiff_hard.ini")

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

5.2 mk_pgpmac_redis Namespace Reference

Variables

- list [head](#) sys.argv[1]
- list [pref_ini](#) sys.argv[2]
- list [hard_ini](#) sys.argv[3]
- dictionary [motor_dict](#)
- dictionary [hard_ini_fields](#)
- list [motor_field_lists](#)
- list [bi_list](#) ["CryoSwitch"]
- dictionary [motor_presets](#)
- list [zoom_settings](#)
- tuple [hi](#) [iniParser.iniParser](#)([hard_ini](#))
- list [v](#) [motor_dict](#)[m]
- string [f](#) "HSETNX"
- tuple [pi](#) [iniParser.iniParser](#)([pref_ini](#))
- int [i](#) 0

- tuple `ppos` `pi.get(section, option)`
- string `fnc` `"HSETNX"`
- tuple `b` `pi.get(section, "LightIntensity")`
- tuple `p` `pi.get(section, "MotorPosition")`
- tuple `x` `pi.get(section, "ScaleX")`
- tuple `y` `pi.get(section, "ScaleY")`

5.2.1 Variable Documentation

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

Definition at line 379 of file `mk_pgpmac_redis.py`.

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

Definition at line 229 of file `mk_pgpmac_redis.py`.

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

Definition at line 329 of file `mk_pgpmac_redis.py`.

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

Definition at line 371 of file `mk_pgpmac_redis.py`.

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

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

5.2.1.6 dictionary `mk_pgpmac_redis.hard_ini_fields`

Initial value:

```

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

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

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

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

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

Definition at line 304 of file `mk_pgpmac_redis.py`.

5.2.1.9 int mk_pgpmac_redis.i 0

Definition at line 350 of file mk_pgpmac_redis.py.

5.2.1.10 dictionary mk_pgpmac_redis.motor_dict

Definition at line 26 of file mk_pgpmac_redis.py.

5.2.1.11 list mk_pgpmac_redis.motor_field_lists

Definition at line 191 of file mk_pgpmac_redis.py.

5.2.1.12 dictionary mk_pgpmac_redis.motor_presets

Definition at line 231 of file mk_pgpmac_redis.py.

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

Definition at line 385 of file mk_pgpmac_redis.py.

5.2.1.14 tuple mk_pgpmac_redis.pi iniParser.iniParser(pref_ini)

Definition at line 347 of file mk_pgpmac_redis.py.

5.2.1.15 tuple mk_pgpmac_redis.ppos pi.get(section, option)

Definition at line 357 of file mk_pgpmac_redis.py.

5.2.1.16 mk_pgpmac_redis.pref_ini sys.argv[2]

Definition at line 16 of file mk_pgpmac_redis.py.

5.2.1.17 tuple mk_pgpmac_redis.v motor_dict[m]

Definition at line 328 of file mk_pgpmac_redis.py.

5.2.1.18 tuple mk_pgpmac_redis.x pi.get(section, "ScaleX")

Definition at line 391 of file mk_pgpmac_redis.py.

5.2.1.19 tuple mk_pgpmac_redis.y pi.get(section, "ScaleY")

Definition at line 397 of file mk_pgpmac_redis.py.

5.2.1.20 list mk_pgpmac_redis.zoom_settings

Initial value:

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

Definition at line 280 of file mk_pgpmac_redis.py.

Chapter 6

Data Structure Documentation

6.1 iniParser.iniParser Class Reference

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

Public Member Functions

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

Data Fields

- `f`
- `sd`

6.1.1 Detailed Description

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

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

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

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

The current version is for READING the files.

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

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

Definition at line 42 of file iniParser.py.

6.1.2 Constructor & Destructor Documentation

6.1.2.1 `def iniParser.iniParser.__init__(self, fn)`

Definition at line 44 of file iniParser.py.

```

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

```

6.1.3 Member Function Documentation

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

Definition at line 99 of file iniParser.py.

```

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

```

6.1.3.2 `def iniParser.iniParser.has_option(self, section, option)`

Definition at line 94 of file iniParser.py.

```

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

```

6.1.3.3 `def iniParser.iniParser.has_section(self, section)`

Definition at line 91 of file iniParser.py.

```

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

```

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

Definition at line 87 of file iniParser.py.

```

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

```


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

Definition at line 49 of file `iniParser.py`.

```

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

```

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

Definition at line 83 of file `iniParser.py`.

```

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

```

6.1.4 Field Documentation

6.1.4.1 `iniParser.iniParser.f`

Definition at line 45 of file `iniParser.py`.

6.1.4.2 `iniParser.iniParser.sd`

Definition at line 46 of file `iniParser.py`.

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

- [iniParser.py](#)

6.2 Isevents_listener_struct Struct Reference

Linked list of event listeners.

Data Fields

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

6.2.1 Detailed Description

Linked list of event listeners.

Definition at line 27 of file lsevents.c.

6.2.2 Field Documentation

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

call back function

Definition at line 31 of file lsevents.c.

6.2.2.2 struct lsevents_listener_struct* lsevents_listener_struct::next

Next listener.

Definition at line 28 of file lsevents.c.

6.2.2.3 char* lsevents_listener_struct::raw_regexp

the original string sent to us

Definition at line 29 of file lsevents.c.

6.2.2.4 regex_t lsevents_listener_struct::re

regular expression representing listened for events

Definition at line 30 of file lsevents.c.

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

- [lsevents.c](#)

6.3 lsevents_queue_struct Struct Reference

Storage definition for the events.

Data Fields

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

6.3.1 Detailed Description

Storage definition for the events.

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

Definition at line 17 of file lsevents.c.

6.3.2 Field Documentation

6.3.2.1 char* lsevents_queue_struct::evp

name of the event

Definition at line 18 of file lsevents.c.

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

- [lsevents.c](#)

6.4 lslogging_queue_struct Struct Reference

Our log object: time and message.

Data Fields

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

6.4.1 Detailed Description

Our log object: time and message.

Definition at line 24 of file lslogging.c.

6.4.2 Field Documentation

6.4.2.1 char lslogging_queue_struct::lmsg[LSLOGGING_MSG_LENGTH]

our message, truncated if too long

Definition at line 26 of file lslogging.c.

6.4.2.2 struct timespec lslogging_queue_struct::itime

time stamp: set when queued

Definition at line 25 of file lslogging.c.

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

- [lslogging.c](#)

6.5 lspg_demandairrights_struct Struct Reference

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

Data Fields

- pthread_mutex_t [mutex](#)
- pthread_cond_t [cond](#)
- int [new_value_ready](#)

6.5.1 Detailed Description

Definition at line 196 of file pgpmac.h.

6.5.2 Field Documentation

6.5.2.1 pthread_cond_t lspg_demandairrights_struct::cond

Definition at line 198 of file pgpmac.h.

6.5.2.2 pthread_mutex_t lspg_demandairrights_struct::mutex

Definition at line 197 of file pgpmac.h.

6.5.2.3 int lspg_demandairrights_struct::new_value_ready

Definition at line 199 of file pgpmac.h.

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

- [pgpmac.h](#)

6.6 lspg_getcenter_struct Struct Reference

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

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

Data Fields

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

6.6.1 Detailed Description

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

Definition at line 210 of file pgpmac.h.

6.6.2 Field Documentation

6.6.2.1 pthread_cond_t lspg_getcenter_struct::cond

provides signaling for when the query is done

Definition at line 212 of file pgpmac.h.

6.6.2.2 double lspg_getcenter_struct::dax

alignment x change

Definition at line 225 of file pgpmac.h.

6.6.2.3 int lspg_getcenter_struct::dax_isnull

Definition at line 226 of file pgpmac.h.

6.6.2.4 double lspg_getcenter_struct::day

alignment y change

Definition at line 228 of file pgpmac.h.

6.6.2.5 int lspg_getcenter_struct::day_isnull

Definition at line 229 of file pgpmac.h.

6.6.2.6 double lspg_getcenter_struct::daz

alignment z change

Definition at line 231 of file pgpmac.h.

6.6.2.7 int lspg_getcenter_struct::daz_isnull

Definition at line 232 of file pgpmac.h.

6.6.2.8 double lspg_getcenter_struct::dcx

center x change

Definition at line 219 of file pgpmac.h.

6.6.2.9 int lspg_getcenter_struct::dcx_isnull

Definition at line 220 of file pgpmac.h.

6.6.2.10 double lspg_getcenter_struct::dcy

center y change

Definition at line 222 of file pgpmac.h.

6.6.2.11 int lspg_getcenter_struct::dcy_isnull

Definition at line 223 of file pgpmac.h.

6.6.2.12 pthread_mutex_t lspg_getcenter_struct::mutex

don't let the threads collide!

Definition at line 211 of file pgpmac.h.

6.6.2.13 int lspg_getcenter_struct::new_value_ready

used with condition

Definition at line 213 of file pgpmac.h.

6.6.2.14 int lspg_getcenter_struct::no_rows_returned

flag in case no centering information was forthcoming

Definition at line 214 of file pgpmac.h.

6.6.2.15 int lspg_getcenter_struct::zoom

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

Definition at line 216 of file pgpmac.h.

6.6.2.16 int lspg_getcenter_struct::zoom_isnull

Definition at line 217 of file pgpmac.h.

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

- [pgpmac.h](#)

6.7 lspg_getcurrentsampleid_struct Struct Reference

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

Data Fields

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

6.7.1 Detailed Description

Definition at line 184 of file pgpmac.h.

6.7.2 Field Documentation

6.7.2.1 pthread_cond_t lspg_getcurrentsampleid_struct::cond

for signaling

Definition at line 186 of file pgpmac.h.

6.7.2.2 unsigned int lspg_getcurrentsampleid_struct::getcurrentsampleid

the sample we think is mounted on the diffractometer

Definition at line 189 of file pgpmac.h.

6.7.2.3 int lspg_getcurrentsampleid_struct::getcurrentsampleid_isnull

the sample we think is mounted on the diffractometer

Definition at line 190 of file pgpmac.h.

6.7.2.4 pthread_mutex_t lspg_getcurrentsampleid_struct::mutex

practice safe threading

Definition at line 185 of file pgpmac.h.

6.7.2.5 int lspg_getcurrentsampleid_struct::new_value_ready

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

Definition at line 188 of file pgpmac.h.

6.7.2.6 int lspg_getcurrentsampleid_struct::no_rows_returned

flag for an empty return

Definition at line 187 of file pgpmac.h.

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

- [pgpmac.h](#)

6.8 lspg_lock_detector_struct Struct Reference

lock detector object Implements detector lock for exposure control

Data Fields

- pthread_mutex_t [mutex](#)
- pthread_cond_t [cond](#)
- int [new_value_ready](#)

6.8.1 Detailed Description

lock detector object Implements detector lock for exposure control

Definition at line 974 of file lspg.c.

6.8.2 Field Documentation

6.8.2.1 pthread_cond_t lspg_lock_detector_struct::cond

Definition at line 976 of file lspg.c.

6.8.2.2 pthread_mutex_t lspg_lock_detector_struct::mutex

Definition at line 975 of file lspg.c.

6.8.2.3 int lspg_lock_detector_struct::new_value_ready

Definition at line 977 of file lspg.c.

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

- [lspg.c](#)

6.9 lspg_lock_diffractionmeter_struct Struct Reference

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

Data Fields

- pthread_mutex_t [mutex](#)
- pthread_cond_t [cond](#)
- int [new_value_ready](#)

6.9.1 Detailed Description

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

Definition at line 915 of file lspg.c.

6.9.2 Field Documentation

6.9.2.1 pthread_cond_t lspg_lock_diffractionmeter_struct::cond

Definition at line 917 of file lspg.c.

6.9.2.2 pthread_mutex_t lspg_lock_diffractionmeter_struct::mutex

Definition at line 916 of file lspg.c.

6.9.2.3 int lspg_lock_diffractionmeter_struct::new_value_ready

Definition at line 918 of file lspg.c.

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

- [lspg.c](#)

6.10 lspg_nextsample_struct Struct Reference

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

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

Data Fields

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

6.10.1 Detailed Description

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

Definition at line 257 of file pgpmac.h.

6.10.2 Field Documentation

6.10.2.1 pthread_cond_t lspg_nextsample_struct::cond

Our condition.

Definition at line 259 of file pgpmac.h.

6.10.2.2 pthread_mutex_t lspg_nextsample_struct::mutex

Our mutex.

Definition at line 258 of file pgpmac.h.

6.10.2.3 int lspg_nextsample_struct::new_value_ready

flag for our condition

Definition at line 260 of file pgpmac.h.

6.10.2.4 unsigned int lspg_nextsample_struct::nextsample

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

Definition at line 263 of file pgpmac.h.

6.10.2.5 int lspg_nextsample_struct::nextsample_isnull

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

Definition at line 264 of file pgpmac.h.

6.10.2.6 int lspg_nextsample_struct::no_rows_returned

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

Definition at line 261 of file pgpmac.h.

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

- [pgpmac.h](#)

6.11 lspg_nextshot_struct Struct Reference

Storage definition for nextshot query.

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

Data Fields

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

- double [dsomega](#)
dataset defined starting omega angle
- int [dsomega_isnull](#)
- double [dskappa](#)
dataset defined starting kappa angle
- int [dskappa_isnull](#)
- double [dsdist](#)
dataset defined detector distance
- int [dsdist_isnull](#)
- double [dsnrg](#)
dataset defined energy
- int [dsnrg_isnull](#)
- unsigned int [dshpid](#)
sample holder ID
- int [dshpid_isnull](#)
- double [cx](#)
centering table x position
- int [cx_isnull](#)
- double [cy](#)
centering table y position
- int [cy_isnull](#)
- double [ax](#)
alignment table x position
- int [ax_isnull](#)
- double [ay](#)
alignment table y position
- int [ay_isnull](#)
- double [az](#)
alignment table z position
- int [az_isnull](#)
- int [active](#)
flag: 1=move to indicated center position, 0=don't move center or alignment tables
- int [active_isnull](#)
- int [sindex](#)
index of frame (used to generate the file extension)
- int [sindex_isnull](#)
- char * [stype](#)
"Normal" or "Gridsearch"
- int [stype_isnull](#)
- double [dsowidth2](#)
next image oscillation width
- int [dsowidth2_isnull](#)
- char * [dsoscaxis2](#)
next image ascillation axis (always "omega")
- int [dsoscaxis2_isnull](#)
- double [dsexp2](#)
next image exposure time
- int [dsexp2_isnull](#)
- double [sstart2](#)
next image start angle
- int [sstart2_isnull](#)
- double [dsphi2](#)

- next image phi position*
 - int [dsphi2_isnull](#)
 - double [dsomega2](#)
- next image omega position*
 - int [dsomega2_isnull](#)
 - double [dskappa2](#)
- next image kappa position*
 - int [dskappa2_isnull](#)
 - double [dsdist2](#)
- next image distance*
 - int [dsdist2_isnull](#)
 - double [dsnrg2](#)
- next image energy*
 - int [dsnrg2_isnull](#)
 - double [cx2](#)
- next image centering table x position*
 - int [cx2_isnull](#)
 - double [cy2](#)
- next image centering table y position*
 - int [cy2_isnull](#)
 - double [ax2](#)
- next image alignment x position*
 - int [ax2_isnull](#)
 - double [ay2](#)
- next image alignment y position*
 - int [ay2_isnull](#)
 - double [az2](#)
- next image alignment z position*
 - int [az2_isnull](#)
 - int [active2](#)
- flag: 1 if next image should use the above centering parameters*
 - int [active2_isnull](#)
 - int [sindex2](#)
- next image index number*
 - int [sindex2_isnull](#)
 - char * [stype2](#)
- next image type ("Normal" or "Gridsearch")*
 - int [stype2_isnull](#)

6.11.1 Detailed Description

Storage definition for nextshot query.

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

Definition at line 277 of file pgpmac.h.

6.11.2 Field Documentation

6.11.2.1 int lspg_nextshot_struct::active

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

Definition at line 340 of file pgpmac.h.

6.11.2.2 `int lspg_nextshot_struct::active2`

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

Definition at line 391 of file pgpmac.h.

6.11.2.3 `int lspg_nextshot_struct::active2_isnull`

Definition at line 392 of file pgpmac.h.

6.11.2.4 `int lspg_nextshot_struct::active_isnull`

Definition at line 341 of file pgpmac.h.

6.11.2.5 `double lspg_nextshot_struct::ax`

alignment table x position

Definition at line 331 of file pgpmac.h.

6.11.2.6 `double lspg_nextshot_struct::ax2`

next image alignment x position

Definition at line 382 of file pgpmac.h.

6.11.2.7 `int lspg_nextshot_struct::ax2_isnull`

Definition at line 383 of file pgpmac.h.

6.11.2.8 `int lspg_nextshot_struct::ax_isnull`

Definition at line 332 of file pgpmac.h.

6.11.2.9 `double lspg_nextshot_struct::ay`

alignment table y position

Definition at line 334 of file pgpmac.h.

6.11.2.10 `double lspg_nextshot_struct::ay2`

next image alignment y position

Definition at line 385 of file pgpmac.h.

6.11.2.11 `int lspg_nextshot_struct::ay2_isnull`

Definition at line 386 of file pgpmac.h.

6.11.2.12 `int lspg_nextshot_struct::ay_isnull`

Definition at line 335 of file pgpmac.h.

6.11.2.13 double lspg_nextshot_struct::az

alignment table z position

Definition at line 337 of file pgpmac.h.

6.11.2.14 double lspg_nextshot_struct::az2

next image alignment z position

Definition at line 388 of file pgpmac.h.

6.11.2.15 int lspg_nextshot_struct::az2_isnull

Definition at line 389 of file pgpmac.h.

6.11.2.16 int lspg_nextshot_struct::az_isnull

Definition at line 338 of file pgpmac.h.

6.11.2.17 pthread_cond_t lspg_nextshot_struct::cond

Condition to wait for a response from our postgresql server.

Definition at line 279 of file pgpmac.h.

6.11.2.18 double lspg_nextshot_struct::cx

centering table x position

Definition at line 325 of file pgpmac.h.

6.11.2.19 double lspg_nextshot_struct::cx2

next image centering table x position

Definition at line 376 of file pgpmac.h.

6.11.2.20 int lspg_nextshot_struct::cx2_isnull

Definition at line 377 of file pgpmac.h.

6.11.2.21 int lspg_nextshot_struct::cx_isnull

Definition at line 326 of file pgpmac.h.

6.11.2.22 double lspg_nextshot_struct::cy

centering table y position

Definition at line 328 of file pgpmac.h.

6.11.2.23 double lspg_nextshot_struct::cy2

next image centering table y position

Definition at line 379 of file pgpmac.h.

6.11.2.24 int lspg_nextshot_struct::cy2_isnull

Definition at line 380 of file pgpmac.h.

6.11.2.25 int lspg_nextshot_struct::cy_isnull

Definition at line 329 of file pgpmac.h.

6.11.2.26 char* lspg_nextshot_struct::dsdir

Directory for data relative to the ESAF home directory.

Definition at line 283 of file pgpmac.h.

6.11.2.27 int lspg_nextshot_struct::dsdir_isnull

Definition at line 284 of file pgpmac.h.

6.11.2.28 double lspg_nextshot_struct::dsdist

dataset defined detector distance

Definition at line 316 of file pgpmac.h.

6.11.2.29 double lspg_nextshot_struct::dsdist2

next image distance

Definition at line 370 of file pgpmac.h.

6.11.2.30 int lspg_nextshot_struct::dsdist2_isnull

Definition at line 371 of file pgpmac.h.

6.11.2.31 int lspg_nextshot_struct::dsdist_isnull

Definition at line 317 of file pgpmac.h.

6.11.2.32 double lspg_nextshot_struct::dsexp

dataset defined exposure time

Definition at line 295 of file pgpmac.h.

6.11.2.33 double lspg_nextshot_struct::dsexp2

next image exposure time

Definition at line 355 of file pgpmac.h.

6.11.2.34 int lspg_nextshot_struct::dsexp2_isnull

Definition at line 356 of file pgpmac.h.

6.11.2.35 int lspg_nextshot_struct::dsexp_isnull

Definition at line 296 of file pgpmac.h.

6.11.2.36 unsigned int lspg_nextshot_struct::dshpid

sample holder ID

Definition at line 322 of file pgpmac.h.

6.11.2.37 int lspg_nextshot_struct::dshpid_isnull

Definition at line 323 of file pgpmac.h.

6.11.2.38 double lspg_nextshot_struct::dskappa

dataset defined starting kappa angle

Definition at line 313 of file pgpmac.h.

6.11.2.39 double lspg_nextshot_struct::dskappa2

next image kappa position

Definition at line 367 of file pgpmac.h.

6.11.2.40 int lspg_nextshot_struct::dskappa2_isnull

Definition at line 368 of file pgpmac.h.

6.11.2.41 int lspg_nextshot_struct::dskappa_isnull

Definition at line 314 of file pgpmac.h.

6.11.2.42 double lspg_nextshot_struct::dsnr

dataset defined energy

Definition at line 319 of file pgpmac.h.

6.11.2.43 `double lspg_nextshot_struct::dsnrg2`

next image energy

Definition at line 373 of file pgpmac.h.

6.11.2.44 `int lspg_nextshot_struct::dsnrg2_isnull`

Definition at line 374 of file pgpmac.h.

6.11.2.45 `int lspg_nextshot_struct::dsnrg_isnull`

Definition at line 320 of file pgpmac.h.

6.11.2.46 `double lspg_nextshot_struct::dsomega`

dataset defined starting omega angle

Definition at line 310 of file pgpmac.h.

6.11.2.47 `double lspg_nextshot_struct::dsomega2`

next image omega position

Definition at line 364 of file pgpmac.h.

6.11.2.48 `int lspg_nextshot_struct::dsomega2_isnull`

Definition at line 365 of file pgpmac.h.

6.11.2.49 `int lspg_nextshot_struct::dsomega_isnull`

Definition at line 311 of file pgpmac.h.

6.11.2.50 `char* lspg_nextshot_struct::dsoscaxis`

dataset defined oscillation axis (always omega)

Definition at line 292 of file pgpmac.h.

6.11.2.51 `char* lspg_nextshot_struct::dsoscaxis2`

next image ascillation axis (always "omega")

Definition at line 352 of file pgpmac.h.

6.11.2.52 `int lspg_nextshot_struct::dsoscaxis2_isnull`

Definition at line 353 of file pgpmac.h.

6.11.2.53 `int lspg_nextshot_struct::dsoscaxis_isnull`

Definition at line 293 of file pgpmac.h.

6.11.2.54 double lspg_nextshot_struct::dsowidth

dataset defined oscillation width

Definition at line 289 of file pgpmac.h.

6.11.2.55 double lspg_nextshot_struct::dsowidth2

next image oscillation width

Definition at line 349 of file pgpmac.h.

6.11.2.56 int lspg_nextshot_struct::dsowidth2_isnull

Definition at line 350 of file pgpmac.h.

6.11.2.57 int lspg_nextshot_struct::dsowidth_isnull

Definition at line 290 of file pgpmac.h.

6.11.2.58 double lspg_nextshot_struct::dsphi

dataset defined starting phi angle

Definition at line 307 of file pgpmac.h.

6.11.2.59 double lspg_nextshot_struct::dsphi2

next image phi position

Definition at line 361 of file pgpmac.h.

6.11.2.60 int lspg_nextshot_struct::dsphi2_isnull

Definition at line 362 of file pgpmac.h.

6.11.2.61 int lspg_nextshot_struct::dsphi_isnull

Definition at line 308 of file pgpmac.h.

6.11.2.62 char* lspg_nextshot_struct::dspid

ID string identifying this dataset.

Definition at line 286 of file pgpmac.h.

6.11.2.63 int lspg_nextshot_struct::dspid_isnull

Definition at line 287 of file pgpmac.h.

6.11.2.64 pthread_mutex_t lspg_nextshot_struct::mutex

Our mutex for sanity in the multi-threaded program.

Definition at line 278 of file pgpmac.h.

6.11.2.65 int lspg_nextshot_struct::new_value_ready

Our flag for the condition to wait for.

Definition at line 280 of file pgpmac.h.

6.11.2.66 int lspg_nextshot_struct::no_rows_returned

flag indicating that no rows were returned.

Definition at line 281 of file pgpmac.h.

6.11.2.67 char* lspg_nextshot_struct::sfn

file name

Definition at line 304 of file pgpmac.h.

6.11.2.68 int lspg_nextshot_struct::sfn_isnull

Definition at line 305 of file pgpmac.h.

6.11.2.69 int lspg_nextshot_struct::sindex

index of frame (used to generate the file extension)

Definition at line 343 of file pgpmac.h.

6.11.2.70 int lspg_nextshot_struct::sindex2

next image index number

Definition at line 394 of file pgpmac.h.

6.11.2.71 int lspg_nextshot_struct::sindex2_isnull

Definition at line 395 of file pgpmac.h.

6.11.2.72 int lspg_nextshot_struct::sindex_isnull

Definition at line 344 of file pgpmac.h.

6.11.2.73 long long lspg_nextshot_struct::skey

key identifying a particular image

Definition at line 298 of file pgpmac.h.

6.11.2.74 `int lspg_nextshot_struct::skey_isnull`

Definition at line 299 of file pgpmac.h.

6.11.2.75 `double lspg_nextshot_struct::sstart`

starting angle

Definition at line 301 of file pgpmac.h.

6.11.2.76 `double lspg_nextshot_struct::sstart2`

next image start angle

Definition at line 358 of file pgpmac.h.

6.11.2.77 `int lspg_nextshot_struct::sstart2_isnull`

Definition at line 359 of file pgpmac.h.

6.11.2.78 `int lspg_nextshot_struct::sstart_isnull`

Definition at line 302 of file pgpmac.h.

6.11.2.79 `char* lspg_nextshot_struct::stype`

"Normal" or "Gridsearch"

Definition at line 346 of file pgpmac.h.

6.11.2.80 `char* lspg_nextshot_struct::stype2`

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

Definition at line 397 of file pgpmac.h.

6.11.2.81 `int lspg_nextshot_struct::stype2_isnull`

Definition at line 398 of file pgpmac.h.

6.11.2.82 `int lspg_nextshot_struct::stype_isnull`

Definition at line 347 of file pgpmac.h.

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

- [pgpmac.h](#)

6.12 lspg_seq_run_prep_struct Struct Reference

Data collection running object.

Data Fields

- pthread_mutex_t [mutex](#)
- pthread_cond_t [cond](#)
- int [new_value_ready](#)

6.12.1 Detailed Description

Data collection running object.

Definition at line 1032 of file lspg.c.

6.12.2 Field Documentation

6.12.2.1 pthread_cond_t lspg_seq_run_prep_struct::cond

Definition at line 1034 of file lspg.c.

6.12.2.2 pthread_mutex_t lspg_seq_run_prep_struct::mutex

Definition at line 1033 of file lspg.c.

6.12.2.3 int lspg_seq_run_prep_struct::new_value_ready

Definition at line 1035 of file lspg.c.

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

- [lspg.c](#)

6.13 lspg_starttransfer_struct Struct Reference

returns 1 if transfer can continue 0 to abort

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

Data Fields

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

6.13.1 Detailed Description

returns 1 if transfer can continue 0 to abort

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

6.13.2 Field Documentation

6.13.2.1 `pthread_cond_t lspg_starttransfer_struct::cond`

Our condition.

Definition at line 245 of file `pgpmac.h`.

6.13.2.2 `pthread_mutex_t lspg_starttransfer_struct::mutex`

Our mutex.

Definition at line 244 of file `pgpmac.h`.

6.13.2.3 `int lspg_starttransfer_struct::new_value_ready`

flag for our condition

Definition at line 246 of file `pgpmac.h`.

6.13.2.4 `int lspg_starttransfer_struct::no_rows_returned`

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

Definition at line 247 of file `pgpmac.h`.

6.13.2.5 `unsigned int lspg_starttransfer_struct::starttransfer`

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

Definition at line 249 of file `pgpmac.h`.

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

- [pgpmac.h](#)

6.14 `lspg_wait_for_detector_struct` Struct Reference

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

Data Fields

- `pthread_mutex_t` [mutex](#)
- `pthread_cond_t` [cond](#)
- `int` [new_value_ready](#)

6.14.1 Detailed Description

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

Definition at line 850 of file lspg.c.

6.14.2 Field Documentation

6.14.2.1 pthread_cond_t lspg_wait_for_detector_struct::cond

Definition at line 852 of file lspg.c.

6.14.2.2 pthread_mutex_t lspg_wait_for_detector_struct::mutex

Definition at line 851 of file lspg.c.

6.14.2.3 int lspg_wait_for_detector_struct::new_value_ready

Definition at line 853 of file lspg.c.

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

- [lspg.c](#)

6.15 lspg_waitcryo_struct Struct Reference

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

Data Fields

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

6.15.1 Detailed Description

Definition at line 176 of file pgpmac.h.

6.15.2 Field Documentation

6.15.2.1 pthread_cond_t lspg_waitcryo_struct::cond

for signaling

Definition at line 178 of file pgpmac.h.

6.15.2.2 pthread_mutex_t Ispg_waitcryo_struct::mutex

practice safe threading

Definition at line 177 of file pgpmac.h.

6.15.2.3 int Ispg_waitcryo_struct::new_value_ready

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

Definition at line 179 of file pgpmac.h.

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

- [pgpmac.h](#)

6.16 IspgQueryQueueStruct Struct Reference

Store each query along with it's callback function.

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

Data Fields

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

6.16.1 Detailed Description

Store each query along with it's callback function.

All calls are asynchronous

Definition at line 31 of file kvredis.c.

6.16.2 Field Documentation

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

Callback function for when a query returns a result.

Definition at line 33 of file kvredis.c.

6.16.2.2 char IspgQueryQueueStruct::qs

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

Definition at line 32 of file kvredis.c.

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

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

6.17 lspmac_ascii_buffers_struct Struct Reference

Data Fields

- uint16_t [command_buf](#)
- uint16_t [command_buf_cc](#)
- char [command_str](#) [160]
- uint16_t [response_buf](#)
- uint16_t [response_n](#)
- char [response_str](#) [256]

6.17.1 Detailed Description

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

6.17.2 Field Documentation

6.17.2.1 uint16_t lspmac_ascii_buffers_struct::command_buf

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

6.17.2.2 uint16_t lspmac_ascii_buffers_struct::command_buf_cc

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

6.17.2.3 char lspmac_ascii_buffers_struct::command_str[160]

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

6.17.2.4 uint16_t lspmac_ascii_buffers_struct::response_buf

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

6.17.2.5 uint16_t lspmac_ascii_buffers_struct::response_n

Definition at line 350 of file `lspmac.c`.

6.17.2.6 char lspmac_ascii_buffers_struct::response_str[256]

Definition at line 351 of file `lspmac.c`.

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

- [lspmac.c](#)

6.18 lspmac_bi_struct Struct Reference

Storage for binary inputs.

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

Data Fields

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

6.18.1 Detailed Description

Storage for binary inputs.

Definition at line 156 of file pgpmac.h.

6.18.2 Field Documentation

6.18.2.1 char* Ispmac_bi_struct::changeEventOff

Event to send when the value changes to 0.

Definition at line 164 of file pgpmac.h.

6.18.2.2 char* Ispmac_bi_struct::changeEventOn

Event to send when the value changes to 1.

Definition at line 163 of file pgpmac.h.

6.18.2.3 int Ispmac_bi_struct::first_time

flag indicating we've not read the input even once

Definition at line 162 of file pgpmac.h.

6.18.2.4 int Ispmac_bi_struct::mask

mask for the bit in the status register

Definition at line 159 of file pgpmac.h.

6.18.2.5 pthread_mutex_t lspmac_bi_struct::mutex

so we don't get confused

Definition at line 158 of file pgpmac.h.

6.18.2.6 int lspmac_bi_struct::position

the current value.

Definition at line 160 of file pgpmac.h.

6.18.2.7 int lspmac_bi_struct::previous

the previous value

Definition at line 161 of file pgpmac.h.

6.18.2.8 int* lspmac_bi_struct::ptr

points to the location in the status buffer

Definition at line 157 of file pgpmac.h.

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

- [pgpmac.h](#)

6.19 lspmac_cmd_queue_struct Struct Reference

PMAC command queue item.

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

Data Fields

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

6.19.1 Detailed Description

PMAC command queue item.

Command queue items are fixed length to simplify memory management.

Definition at line 86 of file pgpmac.h.

6.19.2 Field Documentation

6.19.2.1 `char* lspmac_cmd_queue_struct::event`

event name to send

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

6.19.2.2 `int lspmac_cmd_queue_struct::no_reply`

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

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

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

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

Definition at line 91 of file `pgpmac.h`.

6.19.2.4 `pmac_cmd_t lspmac_cmd_queue_struct::pcmd`

the pmac command to send

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

6.19.2.5 `struct timespec lspmac_cmd_queue_struct::time_sent`

time this item was dequeued and sent to the pmac

Definition at line 89 of file `pgpmac.h`.

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

- [pgpmac.h](#)

6.20 `lspmac_dpascii_queue_struct` Struct Reference

Data Fields

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

6.20.1 Detailed Description

Definition at line 358 of file `lspmac.c`.

6.20.2 Field Documentation

6.20.2.1 `char* lspmac_dpascii_queue_struct::event`

Definition at line 359 of file `lspmac.c`.

6.20.2.2 char lspmac_dpascii_queue_struct::pl[160]

Definition at line 360 of file lspmac.c.

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

- [lspmac.c](#)

6.21 lspmac_motor_struct Struct Reference

Motor information.

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

Data Fields

- pthread_mutex_t [mutex](#)
coordinate waiting for motor to be done
- pthread_cond_t [cond](#)
used to signal when a motor is done moving
- int [not_done](#)
set to 1 when request is queued, zero after motion has toggled
- void(* [read](#))(struct [lspmac_motor_struct](#) *)
method to read the motor status and position
- int [command_sent](#)
Motion command verified sent to pmac.
- int [motion_seen](#)
set to 1 when motion has been verified to have started
- [pmac_cmd_queue_t](#) * [pq](#)
the queue item requesting motion. Used to check time request was made
- int [homing](#)
Homing routine started.
- int [requested_pos_cnts](#)
requested position
- int * [actual_pos_cnts_p](#)
pointer to the md2_status structure to the actual position
- int [actual_pos_cnts](#)
local copy of actual counts so only our mutex is needed to read
- double [position](#)
scaled position
- double [reported_position](#)
previous position reported to the database
- double [requested_position](#)
The position as requested by the user.
- int * [status1_p](#)
First 24 bit PMAC motor status word.
- int [status1](#)
local copy of status1
- int * [status2_p](#)
Second 24 bit PMAC motor status word.
- int [status2](#)

- local copy of status2*
- char * [dac_mvar](#)
controlling mvariable as a string
- char * [name](#)
Name of motor as referred by ls database kvs table.
- [lsredis_obj_t](#) * [active](#)
Use the motor ("true") or not ("false")
- [lsredis_obj_t](#) * [active_init](#)
pmac commands to make this motor active
- [lsredis_obj_t](#) * [axis](#)
the axis (X, Y, Z, etc) or null if not in a coordinate system
- [lsredis_obj_t](#) * [coord_num](#)
coordinate system this motor belongs to (0 if none)
- [lsredis_obj_t](#) * [home](#)
pmac commands to home motor
- [lsredis_obj_t](#) * [inactive_init](#)
pmac commands to inactivate the motor
- [lsredis_obj_t](#) * [max_accel](#)
our maximum acceleration (cts/msec²)
- [lsredis_obj_t](#) * [max_pos](#)
our maximum position (soft limit)
- [lsredis_obj_t](#) * [max_speed](#)
our maximum speed (cts/msec)
- [lsredis_obj_t](#) * [min_pos](#)
our minimum position (soft limit)
- [lsredis_obj_t](#) * [motor_num](#)
pmac motor number
- [lsredis_obj_t](#) * [neutral_pos](#)
zero offset
- [lsredis_obj_t](#) * [pos_limit_hit](#)
positive limit status
- [lsredis_obj_t](#) * [neg_limit_hit](#)
negative limit status
- [lsredis_obj_t](#) * [precision](#)
moves of less than this amount may be ignored
- [lsredis_obj_t](#) * [printf_fmt](#)
printf format
- [lsredis_obj_t](#) * [redis_fmt](#)
special format string to create text array for putting the position back into redis
- [lsredis_obj_t](#) * [redis_position](#)
how we report our position to the world
- [lsredis_obj_t](#) * [status_str](#)
A talky version of the status.
- [lsredis_obj_t](#) * [u2c](#)
conversion from counts to units: 0.0 means not loaded yet
- [lsredis_obj_t](#) * [unit](#)
string to use as the units
- [lsredis_obj_t](#) * [update_resolution](#)
Change needs to be at least this big to report as a new position to the database.
- char * [write_fmt](#)
Format string to write requested position to PMAC used for binary io.

- int * `read_ptr`
With read_mask finds bit to read for binary i/o.
- int `read_mask`
With read_ptr find bit to read for binary i/o.
- int(* `moveAbs`)(struct `lspmac_motor_struct` *, double)
function to move the motor
- double * `lut`
lookup table (instead of u2c)
- int `nlut`
length of lut
- WINDOW * `win`
our ncurses window

6.21.1 Detailed Description

Motor information.

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

Definition at line 101 of file `pgpmac.h`.

6.21.2 Field Documentation

6.21.2.1 `Isredis_obj_t* lspmac_motor_struct::active`

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

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

6.21.2.2 `Isredis_obj_t* lspmac_motor_struct::active_init`

pmac commands to make this motor active

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

6.21.2.3 `int lspmac_motor_struct::actual_pos_cnts`

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

Definition at line 112 of file `pgpmac.h`.

6.21.2.4 `int* lspmac_motor_struct::actual_pos_cnts_p`

pointer to the `md2_status` structure to the actual position

Definition at line 111 of file `pgpmac.h`.

6.21.2.5 `Isredis_obj_t* lspmac_motor_struct::axis`

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

Definition at line 124 of file `pgpmac.h`.

6.21.2.6 `int lspmac_motor_struct::command_sent`

Motion command verified sent to pmac.

Definition at line 106 of file pgpmac.h.

6.21.2.7 `pthread_cond_t lspmac_motor_struct::cond`

used to signal when a motor is done moving

Definition at line 103 of file pgpmac.h.

6.21.2.8 `lsredis_obj_t* lspmac_motor_struct::coord_num`

coordinate system this motor belongs to (0 if none)

Definition at line 125 of file pgpmac.h.

6.21.2.9 `char* lspmac_motor_struct::dac_mvar`

controlling mvariable as a string

Definition at line 120 of file pgpmac.h.

6.21.2.10 `lsredis_obj_t* lspmac_motor_struct::home`

pmac commands to home motor

Definition at line 126 of file pgpmac.h.

6.21.2.11 `int lspmac_motor_struct::homing`

Homing routine started.

Definition at line 109 of file pgpmac.h.

6.21.2.12 `lsredis_obj_t* lspmac_motor_struct::inactive_init`

pmac commands to inactivate the motor

Definition at line 127 of file pgpmac.h.

6.21.2.13 `double* lspmac_motor_struct::lut`

lookup table (instead of u2c)

Definition at line 148 of file pgpmac.h.

6.21.2.14 `lsredis_obj_t* lspmac_motor_struct::max_accel`

our maximum acceleration (cts/msec²)

Definition at line 128 of file pgpmac.h.

6.21.2.15 `lsredis_obj_t* lspmac_motor_struct::max_pos`

our maximum position (soft limit)

Definition at line 129 of file pgpmac.h.

6.21.2.16 `lsredis_obj_t* lspmac_motor_struct::max_speed`

our maximum speed (cts/msec)

Definition at line 130 of file pgpmac.h.

6.21.2.17 `lsredis_obj_t* lspmac_motor_struct::min_pos`

our minimum position (soft limit)

Definition at line 131 of file pgpmac.h.

6.21.2.18 `int lspmac_motor_struct::motion_seen`

set to 1 when motion has been verified to have started

Definition at line 107 of file pgpmac.h.

6.21.2.19 `lsredis_obj_t* lspmac_motor_struct::motor_num`

pmac motor number

Definition at line 132 of file pgpmac.h.

6.21.2.20 `int(* lspmac_motor_struct::moveAbs)(struct lspmac_motor_struct *, double)`

function to move the motor

Definition at line 147 of file pgpmac.h.

6.21.2.21 `pthread_mutex_t lspmac_motor_struct::mutex`

coordinate waiting for motor to be done

Definition at line 102 of file pgpmac.h.

6.21.2.22 `char* lspmac_motor_struct::name`

Name of motor as referred by ls database kvs table.

Definition at line 121 of file pgpmac.h.

6.21.2.23 `lsredis_obj_t* lspmac_motor_struct::neg_limit_hit`

negative limit status

Definition at line 135 of file pgpmac.h.

6.21.2.24 `Isredis_obj_t* lspmac_motor_struct::neutral_pos`

zero offset

Definition at line 133 of file pgpmac.h.

6.21.2.25 `int lspmac_motor_struct::nlut`

length of lut

Definition at line 149 of file pgpmac.h.

6.21.2.26 `int lspmac_motor_struct::not_done`

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

Definition at line 104 of file pgpmac.h.

6.21.2.27 `Isredis_obj_t* lspmac_motor_struct::pos_limit_hit`

positive limit status

Definition at line 134 of file pgpmac.h.

6.21.2.28 `double lspmac_motor_struct::position`

scaled position

Definition at line 113 of file pgpmac.h.

6.21.2.29 `pmac_cmd_queue_t* lspmac_motor_struct::pq`

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

Definition at line 108 of file pgpmac.h.

6.21.2.30 `Isredis_obj_t* lspmac_motor_struct::precision`

moves of less than this amount may be ignored

Definition at line 136 of file pgpmac.h.

6.21.2.31 `Isredis_obj_t* lspmac_motor_struct::printf_fmt`

printf format

Definition at line 137 of file pgpmac.h.

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

method to read the motor status and position

Definition at line 105 of file pgpmac.h.

6.21.2.33 `int lspmac_motor_struct::read_mask`

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

Definition at line 146 of file `pgpmac.h`.

6.21.2.34 `int* lspmac_motor_struct::read_ptr`

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

Definition at line 145 of file `pgpmac.h`.

6.21.2.35 `lsredis_obj_t* lspmac_motor_struct::redis_fmt`

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

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

6.21.2.36 `lsredis_obj_t* lspmac_motor_struct::redis_position`

how we report our position to the world

Definition at line 139 of file `pgpmac.h`.

6.21.2.37 `double lspmac_motor_struct::reported_position`

previous position reported to the database

Definition at line 114 of file `pgpmac.h`.

6.21.2.38 `int lspmac_motor_struct::requested_pos_cnts`

requested position

Definition at line 110 of file `pgpmac.h`.

6.21.2.39 `double lspmac_motor_struct::requested_position`

The position as requested by the user.

Definition at line 115 of file `pgpmac.h`.

6.21.2.40 `int lspmac_motor_struct::status1`

local copy of status1

Definition at line 117 of file `pgpmac.h`.

6.21.2.41 `int* lspmac_motor_struct::status1_p`

First 24 bit PMAC motor status word.

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

6.21.2.42 `int lspmac_motor_struct::status2`

local copy of status2

Definition at line 119 of file pgpmac.h.

6.21.2.43 `int* lspmac_motor_struct::status2_p`

Second 24 bit PMAC motor status word.

Definition at line 118 of file pgpmac.h.

6.21.2.44 `Isredis_obj_t* lspmac_motor_struct::status_str`

A talky version of the status.

Definition at line 140 of file pgpmac.h.

6.21.2.45 `Isredis_obj_t* lspmac_motor_struct::u2c`

conversion from counts to units: 0.0 means not loaded yet

Definition at line 141 of file pgpmac.h.

6.21.2.46 `Isredis_obj_t* lspmac_motor_struct::unit`

string to use as the units

Definition at line 142 of file pgpmac.h.

6.21.2.47 `Isredis_obj_t* lspmac_motor_struct::update_resolution`

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

Definition at line 143 of file pgpmac.h.

6.21.2.48 `WINDOW* lspmac_motor_struct::win`

our ncurses window

Definition at line 150 of file pgpmac.h.

6.21.2.49 `char* lspmac_motor_struct::write_fmt`

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

Definition at line 144 of file pgpmac.h.

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

- [pgpmac.h](#)

6.22 `Isredis_obj_struct` Struct Reference

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

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

Data Fields

- pthread_mutex_t [mutex](#)
Don't let anyone use an old value.
- pthread_cond_t [cond](#)
wait for a valid value
- struct [lsredis_obj_struct](#) * [next](#)
the next in our list (I guess this is going to be a linked list)
- char [valid](#)
1 if we think the value is good, 0 otherwise
- int [wait_for_me](#)
Number of times we need to see our publication before we start accepting new values.
- char * [key](#)
The redis key for this object.
- char * [events_name](#)
Name used to generate events (normally key without the station id)
- int [value_length](#)
Number of bytes allocated for value (not value's string length)
- char * [value](#)
our value
- double [dvalue](#)
our value as a double
- long int [lvalue](#)
our value as a long
- char ** [avalue](#)
our value as an array of strings
- int [bvalue](#)
our value as a boolean (1 or 0) -1 means we couldn't figure it out
- char [cvalue](#)
just the first character of our value
- int [hits](#)
number of times we've searched for this key

6.22.1 Detailed Description

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

Definition at line 38 of file pgpmac.h.

6.22.2 Field Documentation

6.22.2.1 char** [lsredis_obj_struct::avalue](#)

our value as an array of strings

Definition at line 50 of file pgpmac.h.

6.22.2.2 int [lsredis_obj_struct::bvalue](#)

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

Definition at line 51 of file pgpmac.h.

6.22.2.3 pthread_cond_t lsredis_obj_struct::cond

wait for a valid value

Definition at line 40 of file pgpmac.h.

6.22.2.4 char lsredis_obj_struct::cvalue

just the first character of our value

Definition at line 52 of file pgpmac.h.

6.22.2.5 double lsredis_obj_struct::dvalue

our value as a double

Definition at line 48 of file pgpmac.h.

6.22.2.6 char* lsredis_obj_struct::events_name

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

Definition at line 45 of file pgpmac.h.

6.22.2.7 int lsredis_obj_struct::hits

number of times we've searched for this key

Definition at line 53 of file pgpmac.h.

6.22.2.8 char* lsredis_obj_struct::key

The redis key for this object.

Definition at line 44 of file pgpmac.h.

6.22.2.9 long int lsredis_obj_struct::lvalue

our value as a long

Definition at line 49 of file pgpmac.h.

6.22.2.10 pthread_mutex_t lsredis_obj_struct::mutex

Don't let anyone use an old value.

Definition at line 39 of file pgpmac.h.

6.22.2.11 struct lsredis_obj_struct* lsredis_obj_struct::next

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

Definition at line 41 of file pgpmac.h.

6.22.2.12 `char lsredis_obj_struct::valid`

1 if we think the value is good, 0 otherwise

Definition at line 42 of file pgpmac.h.

6.22.2.13 `char* lsredis_obj_struct::value`

our value

Definition at line 47 of file pgpmac.h.

6.22.2.14 `int lsredis_obj_struct::value_length`

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

Definition at line 46 of file pgpmac.h.

6.22.2.15 `int lsredis_obj_struct::wait_for_me`

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

Definition at line 43 of file pgpmac.h.

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

- [pgpmac.h](#)

6.23 `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
- `long int next_secs`
epoch (seconds) of next alarm
- `long int next_nsecs`
nano seconds of next alarm
- `long int delay_secs`
number of seconds for a periodic delay
- `long int delay_nsecs`
nano seconds of delay
- `long int last_secs`
the last time this timer was triggered
- `long int last_nsecs`
the last time this timer was triggered
- `long int init_secs`

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

6.23.1 Detailed Description

Everything we need to know about a timer.

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

6.23.2 Field Documentation

6.23.2.1 long int `Istimer_list_struct::delay_nsecs`

nano seconds of delay

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

6.23.2.2 long int `Istimer_list_struct::delay_secs`

number of seconds for a periodic delay

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

6.23.2.3 char `Istimer_list_struct::event[LSEVENTS_EVENT_LENGTH]`

the event to send

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

6.23.2.4 long int `Istimer_list_struct::init_nsecs`

our initialization time

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

6.23.2.5 long int `Istimer_list_struct::init_secs`

our initialization time

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

6.23.2.6 long int `Istimer_list_struct::last_nsecs`

the last time this timer was triggered

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

6.23.2.7 long int `Istimer_list_struct::last_secs`

the last time this timer was triggered

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

6.23.2.8 unsigned long int `lstimer_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 `lstimer.c`.

6.23.2.9 long int `lstimer_list_struct::next_nsecs`

nano seconds of next alarm

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

6.23.2.10 long int `lstimer_list_struct::next_secs`

epoch (seconds) of next alarm

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

6.23.2.11 int `lstimer_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 `lstimer.c`.

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

- [lstimer.c](#)

6.24 md2StatusStruct Struct Reference

The block of memory retrieved in a status request.

Data Fields

- int [dummy1](#)
- int [omega_status_1](#)
- int [alignx_status_1](#)
- int [aligny_status_1](#)
- int [alignz_status_1](#)
- int [analyzer_status_1](#)
- int [zoom_status_1](#)
- int [aperturey_status_1](#)
- int [aperturez_status_1](#)
- int [copy_status_1](#)
- int [capz_status_1](#)
- int [scint_status_1](#)
- int [centerx_status_1](#)
- int [centery_status_1](#)
- int [kappa_status_1](#)
- int [phi_status_1](#)
- int [dummy2](#)
- int [omega_status_2](#)
- int [alignx_status_2](#)
- int [aligny_status_2](#)

- int [alignz_status_2](#)
- int [analyzer_status_2](#)
- int [zoom_status_2](#)
- int [aperturey_status_2](#)
- int [aperturez_status_2](#)
- int [capy_status_2](#)
- int [capz_status_2](#)
- int [scint_status_2](#)
- int [centerx_status_2](#)
- int [centery_status_2](#)
- int [kappa_status_2](#)
- int [phi_status_2](#)
- int [dummy3](#)
- int [omega_act_pos](#)
- int [alignx_act_pos](#)
- int [aligny_act_pos](#)
- int [alignz_act_pos](#)
- int [analyzer_act_pos](#)
- int [zoom_act_pos](#)
- int [aperturey_act_pos](#)
- int [aperturez_act_pos](#)
- int [capy_act_pos](#)
- int [capz_act_pos](#)
- int [scint_act_pos](#)
- int [centerx_act_pos](#)
- int [centery_act_pos](#)
- int [kappa_act_pos](#)
- int [phi_act_pos](#)
- int [acc11c_1](#)
- int [acc11c_2](#)
- int [acc11c_3](#)
- int [acc11c_5](#)
- int [acc11c_6](#)
- int [front_dac](#)
- int [back_dac](#)
- int [scint_piezo](#)
- int [dummy4](#)
- int [dummy5](#)
- int [dummy6](#)
- int [dummy7](#)
- int [dummy8](#)
- int [dummy9](#)
- int [dummyA](#)
- int [dummyB](#)
- int [fs_is_open](#)
- int [phiscan](#)
- int [fs_has_opened](#)
- int [fs_has_opened_globally](#)
- int [number_passes](#)
- int [moving_flags](#)

6.24.1 Detailed Description

The block of memory retrieved in a status request.

Definition at line 245 of file `lspmac.c`.

6.24.2 Field Documentation

6.24.2.1 `int md2StatusStruct::acc11c_1`

Definition at line 312 of file `lspmac.c`.

6.24.2.2 `int md2StatusStruct::acc11c_2`

Definition at line 313 of file `lspmac.c`.

6.24.2.3 `int md2StatusStruct::acc11c_3`

Definition at line 314 of file `lspmac.c`.

6.24.2.4 `int md2StatusStruct::acc11c_5`

Definition at line 315 of file `lspmac.c`.

6.24.2.5 `int md2StatusStruct::acc11c_6`

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

6.24.2.6 `int md2StatusStruct::alignx_act_pos`

Definition at line 296 of file `lspmac.c`.

6.24.2.7 `int md2StatusStruct::alignx_status_1`

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

6.24.2.8 `int md2StatusStruct::alignx_status_2`

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

6.24.2.9 `int md2StatusStruct::aligny_act_pos`

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

6.24.2.10 `int md2StatusStruct::aligny_status_1`

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

6.24.2.11 `int md2StatusStruct::aligny_status_2`

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

6.24.2.12 `int md2StatusStruct::alignz_act_pos`

Definition at line 298 of file `lspmac.c`.

6.24.2.13 int md2StatusStruct::alignz_status_1

Definition at line 264 of file lspmac.c.

6.24.2.14 int md2StatusStruct::alignz_status_2

Definition at line 281 of file lspmac.c.

6.24.2.15 int md2StatusStruct::analyzer_act_pos

Definition at line 299 of file lspmac.c.

6.24.2.16 int md2StatusStruct::analyzer_status_1

Definition at line 265 of file lspmac.c.

6.24.2.17 int md2StatusStruct::analyzer_status_2

Definition at line 282 of file lspmac.c.

6.24.2.18 int md2StatusStruct::aperturey_act_pos

Definition at line 301 of file lspmac.c.

6.24.2.19 int md2StatusStruct::aperturey_status_1

Definition at line 267 of file lspmac.c.

6.24.2.20 int md2StatusStruct::aperturey_status_2

Definition at line 284 of file lspmac.c.

6.24.2.21 int md2StatusStruct::aperturez_act_pos

Definition at line 302 of file lspmac.c.

6.24.2.22 int md2StatusStruct::aperturez_status_1

Definition at line 268 of file lspmac.c.

6.24.2.23 int md2StatusStruct::aperturez_status_2

Definition at line 285 of file lspmac.c.

6.24.2.24 int md2StatusStruct::back_dac

Definition at line 318 of file lspmac.c.

6.24.2.25 `int md2StatusStruct::copy_act_pos`

Definition at line 303 of file `lspmac.c`.

6.24.2.26 `int md2StatusStruct::copy_status_1`

Definition at line 269 of file `lspmac.c`.

6.24.2.27 `int md2StatusStruct::copy_status_2`

Definition at line 286 of file `lspmac.c`.

6.24.2.28 `int md2StatusStruct::capz_act_pos`

Definition at line 304 of file `lspmac.c`.

6.24.2.29 `int md2StatusStruct::capz_status_1`

Definition at line 270 of file `lspmac.c`.

6.24.2.30 `int md2StatusStruct::capz_status_2`

Definition at line 287 of file `lspmac.c`.

6.24.2.31 `int md2StatusStruct::centerx_act_pos`

Definition at line 306 of file `lspmac.c`.

6.24.2.32 `int md2StatusStruct::centerx_status_1`

Definition at line 272 of file `lspmac.c`.

6.24.2.33 `int md2StatusStruct::centerx_status_2`

Definition at line 289 of file `lspmac.c`.

6.24.2.34 `int md2StatusStruct::centery_act_pos`

Definition at line 307 of file `lspmac.c`.

6.24.2.35 `int md2StatusStruct::centery_status_1`

Definition at line 273 of file `lspmac.c`.

6.24.2.36 `int md2StatusStruct::centery_status_2`

Definition at line 290 of file `lspmac.c`.

6.24.2.37 int md2StatusStruct::dummy1

Definition at line 260 of file lspmac.c.

6.24.2.38 int md2StatusStruct::dummy2

Definition at line 277 of file lspmac.c.

6.24.2.39 int md2StatusStruct::dummy3

Definition at line 294 of file lspmac.c.

6.24.2.40 int md2StatusStruct::dummy4

Definition at line 321 of file lspmac.c.

6.24.2.41 int md2StatusStruct::dummy5

Definition at line 322 of file lspmac.c.

6.24.2.42 int md2StatusStruct::dummy6

Definition at line 323 of file lspmac.c.

6.24.2.43 int md2StatusStruct::dummy7

Definition at line 324 of file lspmac.c.

6.24.2.44 int md2StatusStruct::dummy8

Definition at line 325 of file lspmac.c.

6.24.2.45 int md2StatusStruct::dummy9

Definition at line 326 of file lspmac.c.

6.24.2.46 int md2StatusStruct::dummyA

Definition at line 327 of file lspmac.c.

6.24.2.47 int md2StatusStruct::dummyB

Definition at line 328 of file lspmac.c.

6.24.2.48 int md2StatusStruct::front_dac

Definition at line 317 of file lspmac.c.

6.24.2.49 `int md2StatusStruct::fs_has_opened`

Definition at line 332 of file `lspmac.c`.

6.24.2.50 `int md2StatusStruct::fs_has_opened_globally`

Definition at line 333 of file `lspmac.c`.

6.24.2.51 `int md2StatusStruct::fs_is_open`

Definition at line 330 of file `lspmac.c`.

6.24.2.52 `int md2StatusStruct::kappa_act_pos`

Definition at line 308 of file `lspmac.c`.

6.24.2.53 `int md2StatusStruct::kappa_status_1`

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

6.24.2.54 `int md2StatusStruct::kappa_status_2`

Definition at line 291 of file `lspmac.c`.

6.24.2.55 `int md2StatusStruct::moving_flags`

Definition at line 336 of file `lspmac.c`.

6.24.2.56 `int md2StatusStruct::number_passes`

Definition at line 334 of file `lspmac.c`.

6.24.2.57 `int md2StatusStruct::omega_act_pos`

Definition at line 295 of file `lspmac.c`.

6.24.2.58 `int md2StatusStruct::omega_status_1`

Definition at line 261 of file `lspmac.c`.

6.24.2.59 `int md2StatusStruct::omega_status_2`

Definition at line 278 of file `lspmac.c`.

6.24.2.60 `int md2StatusStruct::phi_act_pos`

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

6.24.2.61 int md2StatusStruct::phi_status_1

Definition at line 275 of file lspmac.c.

6.24.2.62 int md2StatusStruct::phi_status_2

Definition at line 292 of file lspmac.c.

6.24.2.63 int md2StatusStruct::phiscan

Definition at line 331 of file lspmac.c.

6.24.2.64 int md2StatusStruct::scint_act_pos

Definition at line 305 of file lspmac.c.

6.24.2.65 int md2StatusStruct::scint_piezo

Definition at line 319 of file lspmac.c.

6.24.2.66 int md2StatusStruct::scint_status_1

Definition at line 271 of file lspmac.c.

6.24.2.67 int md2StatusStruct::scint_status_2

Definition at line 288 of file lspmac.c.

6.24.2.68 int md2StatusStruct::zoom_act_pos

Definition at line 300 of file lspmac.c.

6.24.2.69 int md2StatusStruct::zoom_status_1

Definition at line 266 of file lspmac.c.

6.24.2.70 int md2StatusStruct::zoom_status_2

Definition at line 283 of file lspmac.c.

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

- [lspmac.c](#)

6.25 tagEthernetCmd Struct Reference

PMAC ethernet packet definition.

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

Data Fields

- unsigned char [RequestType](#)
VR_UPLOAD or VR_DOWNLOAD.
- unsigned char [Request](#)
The command to run (VR_PMAC_GETMEM, etc).
- unsigned short [wValue](#)
Command parameter 1.
- unsigned short [wIndex](#)
Command parameter 2.
- unsigned short [wLength](#)
Number of bytes in bData.
- unsigned char [bData](#) [1492]
The data buffer, if required.

6.25.1 Detailed Description

PMAC ethernet packet definition.

Taken directly from the Delta Tau documentation.

Definition at line 73 of file pgpmac.h.

6.25.2 Field Documentation

6.25.2.1 unsigned char tagEthernetCmd::bData[1492]

The data buffer, if required.

Definition at line 79 of file pgpmac.h.

6.25.2.2 unsigned char tagEthernetCmd::Request

The command to run (VR_PMAC_GETMEM, etc).

Definition at line 75 of file pgpmac.h.

6.25.2.3 unsigned char tagEthernetCmd::RequestType

VR_UPLOAD or VR_DOWNLOAD.

Definition at line 74 of file pgpmac.h.

6.25.2.4 unsigned short tagEthernetCmd::wIndex

Command parameter 2.

Definition at line 77 of file pgpmac.h.

6.25.2.5 unsigned short tagEthernetCmd::wLength

Number of bytes in bData.

Definition at line 78 of file pgpmac.h.

6.25.2.6 unsigned short tagEthernetCmd::wValue

Command parameter 1.

Definition at line 76 of file pgpmac.h.

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

- [pgpmac.h](#)

Chapter 7

File Documentation

7.1 iniParser.py File Reference

Data Structures

- class [iniParser.iniParser](#)

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

Namespaces

- namespace [iniParser](#)

Variables

- tuple [iniParser.jp](#) `iniParser("21-ID-E/microdiff_hard.ini")`

7.2 kvredis.c File Reference

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

Data Structures

- struct [lspgQueryQueueStruct](#)

Store each query along with it's callback function.

Macros

- `#define` [LS_PG_QUERY_QUEUE_LENGTH](#) 512

- `#define LS_PG_QUERY_STRING_LENGTH 512`
- `#define LS_PG_STATE_INIT -4`
- `#define LS_PG_STATE_INIT_POLL -3`
- `#define LS_PG_STATE_RESET -2`
- `#define LS_PG_STATE_RESET_POLL -1`
- `#define LS_PG_STATE_IDLE 1`
- `#define LS_PG_STATE_SEND 2`
- `#define LS_PG_STATE_SEND_FLUSH 3`
- `#define LS_PG_STATE_RECV 4`

Typedefs

- typedef struct `lspgQueryQueueStruct` `lspg_query_queue_t`

Store each query along with it's callback function.

Functions

- void `redisDisconnectCB` (const `redisAsyncContext` *ac, int status)
- void `debugCB` (`redisAsyncContext` *ac, void *reply, void *privdata)
- void `addRead` (void *data)
- void `delRead` (void *data)
- void `addWrite` (void *data)
- void `delWrite` (void *data)
- void `cleanup` (void *data)
- void `lspg_allkvs_cb` (`lspg_query_queue_t` *qqp, `PGresult` *pgr)
- PQnoticeProcessor `lspg_notice_processor` (void *arg, const char *msg)
- `lspg_query_queue_t` * `lspg_query_next` ()
Return the next item in the postgresql queue.
- void `lspg_query_reply_next` ()
Remove the oldest item in the queue.
- `lspg_query_queue_t` * `lspg_query_reply_peek` ()
Return the next item in the reply queue but don't pop it since we may need it more than once.
- void `lspg_query_push` (void(*cb)(`lspg_query_queue_t` *, `PGresult` *), char *fmt,...)
Place a query on the queue.
- void `lspg_receive` ()
Receive a result of a query.
- void `lspg_pg_connect` ()
Connect to the pg server.
- void `lspg_flush` ()
Flush psql output buffer (ie, send the query)
- void `lspg_next_state` ()
Implements our state machine Does not strictly only set the next state as it also calls some functions that, perhaps, alters the state mid-function.
- void `lspg_send_next_query` ()
send the next queued query to the DB server
- void `lspg_pg_service` (struct pollfd *evt)
I/O control to/from the postgresql server.
- void `fd_service` (struct pollfd *evt)
- `main` ()

Variables

- static redisAsyncContext * [subac](#)
- static redisAsyncContext * [cmdac](#)
- static int [ls_pg_state](#) = LS_PG_STATE_INIT
State of the lspg state machine.
- static struct timeval
 [lspg_time_sent](#) [now](#)
used to ensure we do not inundate the db server with connection requests
- static int [kvseq](#) = 0
used to synchronize pg.kvs and redis
- static [lspg_query_queue_t](#) [lspg_query_queue](#) [LS_PG_QUERY_QUEUE_LENGTH]
Our query queue.
- static unsigned int [lspg_query_queue_on](#) = 0
Next position to add something to the queue.
- static unsigned int [lspg_query_queue_off](#) = 0
The last item still being used (on == off means nothing in queue)
- static unsigned int [lspg_query_queue_reply](#) = 0
The current item being digested.
- static PGconn * [q](#) = NULL
Database connector.
- static PostgresPollingStatusType [lspg_connectPoll_response](#)
Used to determine state while connecting.
- static PostgresPollingStatusType [lspg_resetPoll_response](#)
Used to determine state while reconnecting.
- static struct pollfd [lspgfd](#)
our poll info
- static struct pollfd [subfd](#)
poll info for redis subscribe channel
- static struct pollfd [cmdfd](#)
poll info for redis command channel

7.2.1 Macro Definition Documentation

7.2.1.1 #define LS_PG_QUERY_QUEUE_LENGTH 512

Definition at line 12 of file kvredis.c.

7.2.1.2 #define LS_PG_QUERY_STRING_LENGTH 512

Definition at line 13 of file kvredis.c.

7.2.1.3 #define LS_PG_STATE_IDLE 1

Definition at line 19 of file kvredis.c.

7.2.1.4 #define LS_PG_STATE_INIT -4

Definition at line 15 of file kvredis.c.

7.2.1.5 `#define LS_PG_STATE_INIT_POLL -3`

Definition at line 16 of file kvredis.c.

7.2.1.6 `#define LS_PG_STATE_RECV 4`

Definition at line 22 of file kvredis.c.

7.2.1.7 `#define LS_PG_STATE_RESET -2`

Definition at line 17 of file kvredis.c.

7.2.1.8 `#define LS_PG_STATE_RESET_POLL -1`

Definition at line 18 of file kvredis.c.

7.2.1.9 `#define LS_PG_STATE_SEND 2`

Definition at line 20 of file kvredis.c.

7.2.1.10 `#define LS_PG_STATE_SEND_FLUSH 3`

Definition at line 21 of file kvredis.c.

7.2.2 Typedef Documentation

7.2.2.1 `typedef struct lspgQueryQueueStruct lspg_query_queue_t`

Store each query along with it's callback function.

All calls are asynchronous

7.2.3 Function Documentation

7.2.3.1 `void addRead (void * data)`

Definition at line 111 of file kvredis.c.

```

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

```

7.2.3.2 `void addWrite (void * data)`

Definition at line 121 of file kvredis.c.

```

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

```


7.2.3.3 void cleanup (void * *data*)

Definition at line 131 of file kvredis.c.

```

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

```

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

Definition at line 63 of file kvredis.c.

```

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

        r = (redisReply *)reply;

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

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

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

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

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

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

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

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

```

7.2.3.5 void delRead (void * *data*)

Definition at line 116 of file kvredis.c.

```

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

```

7.2.3.6 void delWrite (void * data)

Definition at line 126 of file kvredis.c.

```

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

```

7.2.3.7 void fd_service (struct pollfd * evt)

Definition at line 636 of file kvredis.c.

```

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

```

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

Definition at line 137 of file kvredis.c.

```

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

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

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

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

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

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

```

```

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

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

```

7.2.3.9 void lspg_flush ()

Flush psql output buffer (ie, send the query)

Definition at line 412 of file kvredis.c.

```

{
    int err;

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

        fprintf( stderr, "flush failed: %s\n", PQerrorMessage( q));

        ls_pg_state = LS_PG_STATE_IDLE;
        //
        // We should probably reset the connection and start from scratch.
        // Probably the connection died.
        //
        break;

    case 0:
        // goodness and joy.
        ls_pg_state = LS_PG_STATE_RECV;
        break;

    case 1:
        // more sending to do
        ls_pg_state = LS_PG_STATE_SEND_FLUSH;
        break;
    }
}

```

7.2.3.10 void lspg_next_state ()

Implements our state machine Does not strictly only set the next state as it also calls some functions that, perhaps, alters the state mid-function.

Definition at line 444 of file kvredis.c.

```

{
    //
    // connect to the database
    //
    if( q == NULL ||
        ls_pg_state == LS_PG_STATE_INIT ||
        ls_pg_state == LS_PG_STATE_RESET ||
        ls_pg_state == LS_PG_STATE_INIT_POLL ||
        ls_pg_state == LS_PG_STATE_RESET_POLL)
        lspg_pg_connect( lspgfd);

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

    switch( ls_pg_state) {
    case LS_PG_STATE_INIT_POLL:
        if( lspg_connectPoll_response ==
            PGRES_POLLING_WRITING)
            lspgfd.events = POLLOUT;
        else if( lspg_connectPoll_response ==
            PGRES_POLLING_READING)
            lspgfd.events = POLLIN;
        else
            lspgfd.events = 0;
        break;
    }
}

```

```

case LS_PG_STATE_RESET_POLL:
    if( lspg_resetPoll_response == PGRES_POLLING_WRITING
    )
        lspgfd.events = POLLOUT;
    else if( lspg_resetPoll_response ==
    PGRES_POLLING_READING)
        lspgfd.events = POLLIN;
    else
        lspgfd.events = 0;
    break;

case LS_PG_STATE_IDLE:
case LS_PG_STATE_RECV:
    lspgfd.events = POLLIN;
    break;

case LS_PG_STATE_SEND:
case LS_PG_STATE_SEND_FLUSH:
    lspgfd.events = POLLOUT;
    break;

default:
    lspgfd.events = 0;
}
}

```

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

Definition at line 182 of file kvredis.c.

```

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

```

7.2.3.12 void lspg_pg_connect ()

Connect to the pg server.

Definition at line 325 of file kvredis.c.

```

{
PGresult *pgr;
int wait_interval = 1;
int connection_init = 0;
int i, err;

if( q == NULL)
    ls_pg_state = LS_PG_STATE_INIT;

switch( ls_pg_state) {
case LS_PG_STATE_INIT:

    if( lspg_time_sent.tv_sec != 0) {
        //
        // Reality check: if it's less the about 10 seconds since the last failed
        // attempt
        // the just chill.
        //
        gettimeofday( &now, NULL);
        if( now.tv_sec - lspg_time_sent.tv_sec < 10) {
            return;
        }
    }

    q = PQconnectStart( "dbname=ls user=lsuser hostaddr=10.1.0.3");
    if( q == NULL) {
        fprintf( stderr, "Out of memory (lspg_pg_connect)");
        exit( -1);
    }

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

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

```

```

    }
    err = PQsetnonblocking( q, 1);
    if( err != 0) {
        fprintf( stderr, "Odd, could not set database connection to nonblocking")
        ;
    }

    ls_pg_state = LS_PG_STATE_INIT_POLL;
    lspg_connectPoll_response = PGRES_POLLING_WRITING;
    //
    // set up the connection for poll
    //
    lspgfd.fd = PQsocket( q);
    break;

case LS_PG_STATE_INIT_POLL:
    if( lspg_connectPoll_response ==
        PGRES_POLLING_FAILED) {
        PQfinish( q);
        q = NULL;
        ls_pg_state = LS_PG_STATE_INIT;
    } else if( lspg_connectPoll_response ==
        PGRES_POLLING_OK) {
        PQsetNoticeProcessor( q, (PQnoticeProcessor)lspg_notice_processor
        , NULL);

        ls_pg_state = LS_PG_STATE_IDLE;
    }
    break;

case LS_PG_STATE_RESET:
    err = PQresetStart( q);
    if( err == 0) {
        PQfinish( q);
        q = NULL;
        ls_pg_state = LS_PG_STATE_INIT;
    } else {
        ls_pg_state = LS_PG_STATE_RESET_POLL;
        lspg_resetPoll_response = PGRES_POLLING_WRITING;
    }
    break;

case LS_PG_STATE_RESET_POLL:
    if( lspg_resetPoll_response == PGRES_POLLING_FAILED)
    {
        PQfinish( q);
        q = NULL;
        ls_pg_state = LS_PG_STATE_INIT;
    } else if( lspg_resetPoll_response ==
        PGRES_POLLING_OK) {
        ls_pg_state = LS_PG_STATE_IDLE;
    }
    break;
}
}

```

7.2.3.13 void lspg_pg_service (struct pollfd * evt)

I/O control to/from the postgresql server.

Parameters

in	evt	The pollfd object that we are responding to
----	-----	---

Definition at line 543 of file kvredis.c.

```

    {
//
// Currently just used to check for notifies
// Other socket communication is done synchronously
//

if( evt->revents & POLLIN) {
    int err;

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

```

```

        ls_pg_state = LS_PG_STATE_RESET;
    }
    return;
}

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

//
// if in IDLE or RECV we need to call consumeInput first
//
if( ls_pg_state == LS_PG_STATE_IDLE) {
    err = PQconsumeInput( q);
    if( err != 1) {
        fprintf( stderr, "consume input failed: %s", PQerrorMessage( q));
        ls_pg_state = LS_PG_STATE_RESET;
        return;
    }
}

if( ls_pg_state == LS_PG_STATE_RECV) {
    lspg_receive();
}

//
// Check for notifies regardless of our state
// Push as many requests as we have notifies.
//
{
    PGnotify *pgn;

    while( 1) {
        pgn = PQnotifies( q);
        if( pgn == NULL)
            break;

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

        PQfreemem( pgn);
    }
}

if( evt->revents & POLLOUT) {

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

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

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

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

```

7.2.3.14 lspg_query_queue_t* lspg_query_next()

Return the next item in the postgresql queue.

If there is an item left in the queue then it is returned. Otherwise, NULL is returned.

Definition at line 191 of file kvredis.c.

```

{
    lspg_query_queue_t *rtn;

    if( lspg_query_queue_off == lspg_query_queue_on
    )
        // Queue is empty
        rtn = NULL;
    else {
        rtn = &(lspg_query_queue[ (lspg_query_queue_off
        ++ ) % LS_PG_QUERY_QUEUE_LENGTH]);
    }
    return rtn;
}

```

7.2.3.15 void lspg_query_push (void(*) (lspg_query_queue_t *, PGresult *) cb, char * fmt, ...)

Place a query on the queue.

Parameters

in	<i>cb</i>	Our callback function that deals with the response
in	<i>fmt</i>	Printf style function to generate the query

Definition at line 234 of file kvredis.c.

```

{
    int idx;
    va_list arg_ptr;

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

    idx = lspg_query_queue_on % LS_PG_QUERY_QUEUE_LENGTH
        ;

    va_start( arg_ptr, fmt);
    vsnprintf( lspg_query_queue[idx].qs,
        LS_PG_QUERY_STRING_LENGTH-1, fmt, arg_ptr);
    va_end( arg_ptr);

    lspg_query_queue[idx].qs[LS_PG_QUERY_STRING_LENGTH
        - 1] = 0;
    lspg_query_queue[idx].onResponse = cb;
    lspg_query_queue_on++;
};

```

7.2.3.16 void lspg_query_reply_next()

Remove the oldest item in the queue.

this is called only when there is nothing else to service the reply: this pop does not return anything. We use the ...reply_peek function to return the next item in the reply queue

Definition at line 211 of file kvredis.c.

```

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

```

7.2.3.17 `lspg_query_queue_t*` `lspg_query_reply_peek ()`

Return the next item in the reply queue but don't pop it since we may need it more than once.

Call `lspg_query_reply_next()` when done.

Definition at line 221 of file kvredis.c.

```

    {
        lspg_query_queue_t *rtn;

        if( lspg_query_queue_reply == lspg_query_queue_on
        )
            rtn = NULL;
        else
            rtn = &(lspg_query_queue[ (lspg_query_queue_reply
            ) % LS_PG_QUERY_QUEUE_LENGTH]);

        return rtn;
    }

```

7.2.3.18 `void` `lspg_receive ()`

Receive a result of a query.

Definition at line 266 of file kvredis.c.

```

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

        err = PQconsumeInput( q );
        if( err != 1 ) {
            fprintf( stderr, "consume input failed: %s", PQerrorMessage( q ));
            ls_pg_state == LS_PG_STATE_RESET;
            return;
        }

        //
        // We must call PQgetResult until it returns NULL before sending the next
        // query
        // This implies that only one query can ever be active at a time and our
        // queue
        // management should be simple
        //
        // We should be in the LS_PG_STATE_RECV here
        //

        while( !PQisBusy( q) ) {
            pgr = PQgetResult( q );
            if( pgr == NULL ) {
                lspg_query_reply_next();
                //
                // we are now done reading the response from the database
                //
                ls_pg_state = LS_PG_STATE_IDLE;
                break;
            } else {
                ExecStatusType es;

                qqp = lspg_query_reply_peek();
                es = PQresultStatus( pgr );

                if( es != PGRES_COMMAND_OK && es != PGRES_TUPLES_OK ) {

```



```

    char *emess;
    emess = PQresultErrorMessage( pgr);
    if( emess != NULL && emess[0] != 0) {
        fprintf( stderr, "Error from query '%s':\n%s", qqp->qs, emess);
    }
    else {
        //
        // Deal with the response
        //
        // If the response is likely to take awhile we should probably
        // add a new state and put something in the main look to run the
        onResponse
        // routine in the main loop. For now, though, we only expect very
        brief onResponse routines
        //
        if( qqp != NULL && qqp->onResponse != NULL)
            qqp->onResponse( qqp, pgr);
    }
    PQclear( pgr);
}
}
}

```

7.2.3.19 void lspg_send_next_query ()

send the next queued query to the DB server

Definition at line 496 of file kvredis.c.

```

{
    //
    // Normally we should be in the "send" state
    // but we can also send if we are servicing
    // a reply
    //

    lspg_query_queue_t *qqp;
    int err;

    qqp = lspg_query_next();
    if( qqp == NULL) {
        //
        // A send without a query? Should never happen.
        // But at least we shouldn't segfault if it does.
        //
        return;
    }

    if( qqp->qs[0] == 0) {
        //
        // Do we really have to check this case?
        // It would only come up if we stupidly pushed an empty query string
        // or ran off the end of the queue
        //
        fprintf( stderr, "Popped empty query string. Probably bad things are going
            on.\n");

        lspg_query_reply_next();
        ls_pg_state = LS_PG_STATE_IDLE;
    } else {
        err = PQsendQuery( q, qqp->qs);
        if( err == 0) {
            fprintf( stderr, "query failed: %s\n", PQerrorMessage( q));

            //
            // Don't wait for a reply, just reset the connection
            //
            lspg_query_reply_next();
            ls_pg_state == LS_PG_STATE_RESET;
        } else {
            ls_pg_state = LS_PG_STATE_SEND_FLUSH;
        }
    }
}
}

```

7.2.3.20 main ()

Definition at line 655 of file kvredis.c.

```

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

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

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

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

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

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

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

lspgfd.fd = -1;

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

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

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

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

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

        nfda++;
    }
    poll_timeout_ms = -1;

```

```

lspg_next_state();

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

pollrtn = poll( fda, nfda, poll_timeout_ms);

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

```

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

Definition at line 54 of file kvredis.c.

```

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

```

7.2.4 Variable Documentation

7.2.4.1 redisAsyncContext * cmdac [static]

Definition at line 9 of file kvredis.c.

7.2.4.2 struct pollfd cmdfd [static]

poll info for redis command channel

Definition at line 50 of file kvredis.c.

7.2.4.3 int kvseq = 0 [static]

used to synchronize pg.kvs and redis

Definition at line 26 of file kvredis.c.

7.2.4.4 int ls_pg_state = LS_PG_STATE_INIT [static]

State of the lspg state machine.

Definition at line 24 of file kvredis.c.

7.2.4.5 `PostgresPollingStatusType lspg_connectPoll_response` `[static]`

Used to determine state while connecting.

Definition at line 46 of file kvredis.c.

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

Our query queue.

Definition at line 37 of file kvredis.c.

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

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

Definition at line 39 of file kvredis.c.

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

Next position to add something to the queue.

Definition at line 38 of file kvredis.c.

7.2.4.9 `unsigned int lspg_query_queue_reply = 0` `[static]`

The current item being digested.

Normally off <= reply <= on. Corner case of queue wrap around works because we only increment and compare for equality.

Definition at line 40 of file kvredis.c.

7.2.4.10 `PostgresPollingStatusType lspg_resetPoll_response` `[static]`

Used to determine state while reconnecting.

Definition at line 47 of file kvredis.c.

7.2.4.11 `struct pollfd lspgfd` `[static]`

our poll info

Definition at line 48 of file kvredis.c.

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

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

Definition at line 25 of file kvredis.c.

7.2.4.13 PGconn* q = NULL [static]

Database connector.

Definition at line 45 of file kvredis.c.

7.2.4.14 redisAsyncContext* subac [static]

Definition at line 9 of file kvredis.c.

7.2.4.15 struct pollfd subfd [static]

poll info for redis subscribe channel

Definition at line 49 of file kvredis.c.

7.3 lsevents.c File Reference

event subsystem for inter-pgpmac communication

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

Data Structures

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

Macros

- #define [LSEVENTS_QUEUE_LENGTH](#) 512

Typedefs

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

Functions

- void [lsevents_send_event](#) (char *fmt,...)
Call the callback routines for the given event.
- void [lsevents_add_listener](#) (char *event, void(*cb)(char *))
Add a callback routine to listen for a specific event.
- void [lsevents_remove_listener](#) (char *event, void(*cb)(char *))
Remove a listener previously added with lsevents_add_listener.

- void * [lsevents_worker](#) (void *dummy)
Our worker.
- void [lsevents_init](#) ()
Initialize this module.
- void [lsevents_run](#) ()
Start up the thread and get out of the way.

Variables

- static [lsevents_queue_t](#) [lsevents_queue](#) [[LSEVENTS_QUEUE_LENGTH](#)]
simple list of events
- static unsigned int [lsevents_queue_on](#) = 0
next queue location to write
- static unsigned int [lsevents_queue_off](#) = 0
next queue location to read
- static [lsevents_listener_t](#) * [lsevents_listeners_p](#) = NULL
Pointer to the first item in the link list of listeners.
- static pthread_t [lsevents_thread](#)
thread to run the event queue
- static pthread_mutex_t [lsevents_listener_mutex](#)
mutex to protect the listener linked list
- static pthread_mutex_t [lsevents_queue_mutex](#)
mutex to protect the event queue
- static pthread_cond_t [lsevents_queue_cond](#)
condition to pause the queue if needed

7.3.1 Detailed Description

event subsystem for inter-pgpmac communication

Date

2012

Author

Keith Brister

Copyright

All Rights Reserved

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

7.3.2 Macro Definition Documentation

7.3.2.1 #define LSEVENTS_QUEUE_LENGTH 512

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

7.3.3 Typedef Documentation

7.3.3.1 typedef struct lsevents_listener_struct lsevents_listener_t

Linked list of event listeners.

7.3.3.2 typedef struct lsevents_queue_struct lsevents_queue_t

Storage definition for the events.

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

7.3.4 Function Documentation

7.3.4.1 void lsevents_add_listener (char * event, void(*)(char *) cb)

Add a callback routine to listen for a specific event.

Parameters

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

Definition at line 75 of file lsevents.c.

```

lsevents_listener_t *new;
int err;
char *errbuf;
int nerrbuf;

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

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

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

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

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

```

7.3.4.2 void lsevents_init ()

Initialize this module.

Definition at line 214 of file lsevents.c.

```

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

```

7.3.4.3 void lsevents_remove_listener (char * event, void(*)(char *) cb)

Remove a listener previously added with lsevents_add_listener.

Parameters

<i>event</i>	The name of the event
<i>cb</i>	The callback routine to remove

Definition at line 120 of file lsevents.c.

```

{
    lsevents_listener_t *last, *current;

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

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

```

7.3.4.4 void lsevents_run ()

Start up the thread and get out of the way.

Definition at line 222 of file lsevents.c.

```

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

```


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

Call the callback routines for the given event.

Parameters

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

Definition at line 45 of file lsevents.c.

```

{
    char event[LSEVENTS_EVENT_LENGTH];
    va_list arg_ptr;

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

    pthread_mutex_lock( &lsevents_queue_mutex);

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

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

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

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

```

7.3.4.6 void* lsevents_worker (void * *dummy*)

Our worker.

Parameters

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

Definition at line 155 of file lsevents.c.

```

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

    while( 1) {
        pthread_mutex_lock( &lsevents_queue_mutex);

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

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

```

```

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

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

pthread_mutex_unlock( &lsevents_listener_mutex);
}
return NULL;
}

```

7.3.5 Variable Documentation

7.3.5.1 pthread_mutex_t lsevents_listener_mutex [static]

mutex to protect the listener linked list

Definition at line 37 of file lsevents.c.

7.3.5.2 lsevents_listener_t* lsevents_listeners_p = NULL [static]

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

Definition at line 34 of file lsevents.c.

7.3.5.3 lsevents_queue_t lsevents_queue[LSEVENTS_QUEUE_LENGTH] [static]

simple list of events

Definition at line 21 of file lsevents.c.

7.3.5.4 pthread_cond_t lsevents_queue_cond [static]

condition to pause the queue if needed

Definition at line 39 of file lsevents.c.

7.3.5.5 pthread_mutex_t lsevents_queue_mutex [static]

mutex to protect the event queue

Definition at line 38 of file lsevents.c.

7.3.5.6 unsigned int lsevents_queue_off = 0 [static]

next queue location to read

Definition at line 23 of file lsevents.c.

7.3.5.7 unsigned int lsevents_queue_on = 0 [static]

next queue location to write

Definition at line 22 of file lsevents.c.

7.3.5.8 pthread_t lsevents_thread [static]

thread to run the event queue

Definition at line 36 of file lsevents.c.

7.4 Islogging.c File Reference

Logs messages to a file.

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

Data Structures

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

Macros

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

Typedefs

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

Functions

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

Service the queue, write to the file.

- void [lslogging_run](#) ()

Start up the worker thread.

Variables

- static pthread_t [lslogging_thread](#)
our thread
- static pthread_mutex_t [lslogging_mutex](#)
mutex to keep the various threads from adding to the queue at the exact same time
- static pthread_cond_t [lslogging_cond](#)
We'll spend most of our time waiting for this condition's signal.
- static FILE * [lslogging_file](#)
our log file object
- static [lslogging_queue_t](#) [lslogging_queue](#) [[LSLOGGING_QUEUE_LENGTH](#)]
Our entire queue. Right here. Every message we'll ever write.
- static unsigned int [lslogging_on](#) = 0
next location to add to the queue
- static unsigned int [lslogging_off](#) = 0
next location to remove from the queue

7.4.1 Detailed Description

Logs messages to a file.

Date

2012

Author

Keith Brister

Copyright

All Rights Reserved

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

7.4.2 Macro Definition Documentation

7.4.2.1 #define LSLOGGING_FILE_NAME "/tmp/pgpmac.log"

Full name of the log file.

Probably should be in /var/log/pgpmac.

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

7.4.2.2 #define LSLOGGING_MSG_LENGTH 2048

Fixed maximum length messages to keep some form of sanity.

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

7.4.2.3 #define LSLOGGING_QUEUE_LENGTH 8192

Modest length queue.

Definition at line 30 of file Islogging.c.

7.4.3 Typedef Documentation

7.4.3.1 typedef struct Islogging_queue_struct Islogging_queue_t

Our log object: time and message.

7.4.4 Function Documentation

7.4.4.1 void Islogging_init ()

Initialize the Islogging objects.

Definition at line 37 of file Islogging.c.

```

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

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

```

7.4.4.2 void Islogging_log_message (char * fmt, ...)

The routine everyone will be talking about.

Parameters

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

Definition at line 48 of file Islogging.c.

```

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

    clock_gettime( CLOCK_REALTIME, &theTime);

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

    pthread_mutex_lock( &lslogging_mutex);

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

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

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

```

7.4.4.3 void lslogging_run ()

Start up the worker thread.

Definition at line 105 of file lslogging.c.

```

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

```

7.4.4.4 void* lslogging_worker (void * dummy)

Service the queue, write to the file.

Parameters

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

Definition at line 76 of file lslogging.c.

```

    {

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

        pthread_mutex_lock( &lslogging_mutex);

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

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

```

7.4.5 Variable Documentation

7.4.5.1 pthread_cond_t lslogging_cond [static]

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

Definition at line 12 of file lslogging.c.

7.4.5.2 FILE* lslogging_file [static]

our log file object

Definition at line 17 of file lslogging.c.

7.4.5.3 pthread_mutex_t lslogging_mutex [static]

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

Definition at line 11 of file lslogging.c.

7.4.5.4 unsigned int lslogging_off = 0 [static]

next location to remove from the queue

Definition at line 34 of file lslogging.c.

7.4.5.5 unsigned int lslogging_on = 0 [static]

next location to add to the queue

Definition at line 33 of file lslogging.c.

7.4.5.6 lslogging_queue_t lslogging_queue[LSLOGGING_QUEUE_LENGTH] [static]

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

Definition at line 31 of file lslogging.c.

7.4.5.7 pthread_t lslogging_thread [static]

our thread

Definition at line 10 of file lslogging.c.

7.5 lspg.c File Reference

Postgresql support for the LS-CAT pgpmac project.

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

Data Structures

- struct [lspg_wait_for_detector_struct](#)
Object that implements detector / spindle timing We use database locks for exposure control and this implements the md2 portion of this handshake.
- struct [lspg_lock_diffractionmeter_struct](#)
Object used to impliment locking the diffractometer Critical to exposure timing.
- struct [lspg_lock_detector_struct](#)
lock detector object Implements detector lock for exposure control
- struct [lspg_seq_run_prep_struct](#)
Data collection running object.

Macros

- #define [LS_PG_STATE_INIT](#) -4
- #define [LS_PG_STATE_INIT_POLL](#) -3
- #define [LS_PG_STATE_RESET](#) -2

- `#define LS_PG_STATE_RESET_POLL -1`
- `#define LS_PG_STATE_IDLE 1`
- `#define LS_PG_STATE_SEND 2`
- `#define LS_PG_STATE_SEND_FLUSH 3`
- `#define LS_PG_STATE_RECV 4`
- `#define LS_PG_QUERY_QUEUE_LENGTH 16384`

Queue length should be long enough that we do not ordinarily bump into the end We should be safe as long as the thread the adds stuff to the queue is not the one that removes it.

Typedefs

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

Functions

- `lspg_query_queue_t * lspg_query_next ()`
Return the next item in the postgresql queue.
- void `lspg_query_reply_next ()`
Remove the oldest item in the queue.
- `lspg_query_queue_t * lspg_query_reply_peek ()`
Return the next item in the reply queue but don't pop it since we may need it more than once.
- void `lspg_query_push (void(*cb)(lspg_query_queue_t *, PGresult *), char *fmt,...)`
Place a query on the queue.
- char ** `lspg_array2ptrs (char *a)`
returns a null terminated list of strings parsed from postgresql array
- void `lspg_starttransfer_init ()`
- void `lspg_starttransfer_cb (lspg_query_queue_t *qqp, PGresult *pgr)`
- void `lspg_starttransfer_call (unsigned int nextsample, int sample_detected, double ax, double ay, double az, double horz, double vert, double esttime)`
- void `lspg_starttransfer_wait ()`
- void `lspg_starttransfer_done ()`
- int `lspg_starttransfer_all (int *err, unsigned int nextsample, int sampledetected, double ax, double ay, double az, double horz, double vert, double esttime)`
- void `lspg_getcurrentsampleid_init ()`
- void `lspg_getcurrentsampleid_cb (lspg_query_queue_t *qqp, PGresult *pgr)`
get currentsampleid
- void `lspg_getcurrentsampleid_call ()`
- unsigned int `lspg_getcurrentsampleid_read ()`
- void `lspg_getcurrentsampleid_wait_for_id (unsigned int test)`
- void `lspg_nextsample_cb (lspg_query_queue_t *qqp, PGresult *pgr)`

- *Next Sample.*
- void [lspg_nextsample_init](#) ()
Initialize the nextsample variable, mutex, and condition.
- void [lspg_nextsample_call](#) ()
Queue up a nextsample query.
- void [lspg_nextsample_wait](#) ()
Wait for the nextsample query to get processed.
- void [lspg_nextsample_done](#) ()
Called when the next shot query has been processed.
- unsigned int [lspg_nextsample_all](#) (int *err)
- void [lspg_waitcryo_init](#) ()
- void [lspg_waitcryo_cb](#) (lspg_query_queue_t *qqp, PGresult *pgr)
- void [lspg_waitcryo_all](#) ()
no need to get fancy with the wait cryo command It should not return until the robot is almost ready for air rights
- void [lspg_demandairrights_init](#) ()
initialize the demandairrights structure
- void [lspg_demandairrights_cb](#) (lspg_query_queue_t *qqp, PGresult *pgr)
handle the airrights response
- void [lspg_demandairrights_call](#) ()
call for airrights
- void [lspg_demandairrights_wait](#) ()
wait for the air rights request to return
- void [lspg_demandairrights_all](#) ()
do nothing until we get airrights
- void [lspg_nextshot_cb](#) (lspg_query_queue_t *qqp, PGresult *pgr)
Next Shot Callback.
- void [lspg_nextshot_init](#) ()
Initialize the nextshot variable, mutex, and condition.
- void [lspg_nextshot_call](#) ()
Queue up a nextshot query.
- void [lspg_nextshot_wait](#) ()
Wait for the next shot query to get processed.
- void [lspg_nextshot_done](#) ()
Called when the next shot query has been processed.
- void [lspg_wait_for_detector_init](#) ()
initialize the detector timing object
- void [lspg_wait_for_detector_cb](#) (lspg_query_queue_t *qqp, PGresult *pgr)
Callback for the wait for detector query.
- void [lspg_wait_for_detector_call](#) ()
initiate the wait for detector query
- void [lspg_wait_for_detector_wait](#) ()
Pause the calling thread until the detector is ready Called by the MD2 thread.
- void [lspg_wait_for_detector_done](#) ()
Done waiting for the detector.
- void [lspg_wait_for_detector_all](#) ()
Combined call to wait for the detector.
- void [lspg_lock_diffractionmeter_init](#) ()
initialize the diffractionmeter locking object
- void [lspg_lock_diffractionmeter_cb](#) (lspg_query_queue_t *qqp, PGresult *pgr)
Callback routine for a lock diffractionmeter query.
- void [lspg_lock_diffractionmeter_call](#) ()

- Request that the database grab the diffractometer lock.*

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

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

 - void `lspg_lock_diffractometer_all ()`
- Convenience function that combines lock diffractometer calls.*

 - void `lspg_lock_detector_init ()`
- Initialize detector lock object.*

 - void `lspg_lock_detector_cb (lspg_query_queue_t *qqp, PGresult *pgr)`
- Callback for when the detector lock has be grabbed.*

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

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

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

 - void `lspg_lock_detector_all ()`
- Detector lock convenience function.*

 - void `lspg_seq_run_prep_init ()`
- Initialize the data collection object.*

 - void `lspg_seq_run_prep_cb (lspg_query_queue_t *qqp, PGresult *pgr)`
- Callback for the seq_run_prep query.*

 - void `lspg_seq_run_prep_call (long long skey, double kappa, double phi, double cx, double cy, double ax, double ay, double az)`
- queue up the seq_run_prep query*

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

 - void `lspg_seq_run_prep_done ()`
- Indicate we are done waiting.*

 - void `lspg_seq_run_prep_all (long long skey, double kappa, double phi, double cx, double cy, double ax, double ay, double az)`
- Convenience function to call seq run prep.*

 - void `lspg_getcenter_cb (lspg_query_queue_t *qqp, PGresult *pgr)`
- Retrieve the data to center the crystal.*

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

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

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

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

 - void `lspg_getcenter_all ()`
- Convenience function to complete synchronous getcenter query.*

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

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

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

- void [lspg_send_next_query](#) ()
send the next queued query to the DB server
- void [lspg_receive](#) ()
Receive a result of a query.
- void [lspg_sig_service](#) (struct pollfd *evt)
Service a signal Signals here are treated as file descriptors and fits into our poll scheme.
- void [lspg_pg_service](#) (struct pollfd *evt)
I/O control to/from the postgresql server.
- PQnoticeProcessor [lspg_notice_processor](#) (void *arg, const char *msg)
- void [lspg_pg_connect](#) ()
Connect to the pg server.
- void [lspg_next_state](#) ()
Implements our state machine Does not strictly only set the next state as it also calls some functions that, perhaps, alters the state mid-function.
- void * [lspg_worker](#) (void *dummy)
The main loop for the lspg thread.
- void [lspmac_sample_detector_cb](#) (char *event)
log magnet state
- void [lspg_init](#) ()
Initialize the lspg module.
- void [lspg_run](#) ()
Start 'er runnin'.

Variables

- static int [ls_pg_state](#) = LS_PG_STATE_INIT
State of the lspg state machine.
- static struct timeval
[lspg_time_sent](#) [now](#)
used to ensure we do not inundate the db server with connection requests
- static pthread_t [lspg_thread](#)
our worker thread
- static pthread_mutex_t [lspg_queue_mutex](#)
keep the queue from getting tangled
- static pthread_cond_t [lspg_queue_cond](#)
keeps the queue from overflowing
- static struct pollfd [lspgfd](#)
our poll info
- static [lspg_query_queue_t](#) [lspg_query_queue](#) [[LS_PG_QUERY_QUEUE_LENGTH](#)]
Our query queue.
- static unsigned int [lspg_query_queue_on](#) = 0
Next position to add something to the queue.
- static unsigned int [lspg_query_queue_off](#) = 0
The last item still being used (on == off means nothing in queue)
- static unsigned int [lspg_query_queue_reply](#) = 0
The current item being digested.
- static PGconn * [q](#) = NULL
Database connector.
- static PostgresPollingStatusType [lspg_connectPoll_response](#)
Used to determine state while connecting.
- static PostgresPollingStatusType [lspg_resetPoll_response](#)

- Used to determine state while reconnecting.*
- [lspg_nextsample_t](#) [lspg_nextsample](#)
the very next sample
- [lspg_nextshot_t](#) [lspg_nextshot](#)
the nextshot object
- [lspg_getcenter_t](#) [lspg_getcenter](#)
the getcenter object
- [lspg_demandairrights_t](#) [lspg_demandairrights](#)
our demandairrights object
- [lspg_getcurrentsampleid_t](#) [lspg_getcurrentsampleid](#)
our currentsample id
- [lspg_starttransfer_t](#) [lspg_starttransfer](#)
start a sample transfer
- [lspg_waitcryo_t](#) [lspg_waitcryo](#)
signal the robot
- static [lspg_wait_for_detector_t](#) [lspg_wait_for_detector](#)
Instance of the detector timing object.
- static [lspg_lock_diffractionmeter_t](#) [lspg_lock_diffractionmeter](#)
- static [lspg_lock_detector_t](#) [lspg_lock_detector](#)
- static [lspg_seq_run_prep_t](#) [lspg_seq_run_prep](#)

7.5.1 Detailed Description

Postgresql support for the LS-CAT pgpmac project.

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

Database state machine

State	Description
-4	Initiate connection
-3	Poll until connection initialization is complete
-2	Initiate reset
-1	Poll until connection reset is complete
1	Idle (wait for a notify from the server)
2	Send a query to the server
3	Continue flushing a command to the server
4	Waiting for a reply

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

7.5.2 Macro Definition Documentation

7.5.2.1 #define LS_PG_QUERY_QUEUE_LENGTH 16384

Queue length should be long enough that we do not ordinarily bump into the end We should be safe as long as the thread the adds stuff to the queue is not the one that removes it.

(And we can tolerate the adding thread being paused.)

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

7.5.2.2 `#define LS_PG_STATE_IDLE 1`

Definition at line 34 of file lspg.c.

7.5.2.3 `#define LS_PG_STATE_INIT -4`

Definition at line 30 of file lspg.c.

7.5.2.4 `#define LS_PG_STATE_INIT_POLL -3`

Definition at line 31 of file lspg.c.

7.5.2.5 `#define LS_PG_STATE_RECV 4`

Definition at line 37 of file lspg.c.

7.5.2.6 `#define LS_PG_STATE_RESET -2`

Definition at line 32 of file lspg.c.

7.5.2.7 `#define LS_PG_STATE_RESET_POLL -1`

Definition at line 33 of file lspg.c.

7.5.2.8 `#define LS_PG_STATE_SEND 2`

Definition at line 35 of file lspg.c.

7.5.2.9 `#define LS_PG_STATE_SEND_FLUSH 3`

Definition at line 36 of file lspg.c.

7.5.3 Typedef Documentation

7.5.3.1 `typedef struct lspg_lock_detector_struct lspg_lock_detector_t`

lock detector object Implements detector lock for exposure control

7.5.3.2 `typedef struct lspg_lock_diffractionmeter_struct lspg_lock_diffractionmeter_t`

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

7.5.3.3 `typedef struct lspg_seq_run_prep_struct lspg_seq_run_prep_t`

Data collection running object.

7.5.3.4 `typedef struct lspg_wait_for_detector_struct lspg_wait_for_detector_t`

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

7.5.4 Function Documentation

7.5.4.1 char** lspg_array2ptrs (char * a)

returns a null terminated list of strings parsed from postgresql array

Definition at line 161 of file lspg.c.

```

{
    char **rtn, *sp, *acums;
    int i, n, inquote, havebackslash, rtni;;
    int mxsz;

    inquote      = 0;
    havebackslash = 0;

    // Despense with the null input condition before we complicate the code below
    if( a == NULL || a[0] != '{' || a[strlen(a)-1] != '}' )
        return NULL;

    // Count the maximum number of strings
    // Actual number will be less if there are quoted commas
    //
    n = 1;
    for( i=0; a[i]; i++) {
        if( a[i] == ',' )
            n++;
    }
    //
    // The maximum size of any string is the length of a (+1)
    //
    mxsz = strlen(a) + 1;

    // This is the accumulation string to make up the array elements
    acums = (char *)calloc( mxsz, sizeof( char));
    if( acums == NULL) {
        lslogging_log_message( "lspg_array2ptrs: out of memory
            (acums)");
        exit( 1);
    }

    //
    // allocate storage for the pointer array and the null terminator
    //
    rtn = (char **)calloc( n+1, sizeof( char *));
    if( rtn == NULL) {
        lslogging_log_message( "lspg_array2ptrs: out of memory
            (rtn)");
        exit( 1);
    }
    rtni = 0;

    // Go through and create the individual strings
    sp = acums;
    *sp = 0;

    inquote = 0;
    havebackslash = 0;
    for( i=1; a[i] != 0; i++) {
        switch( a[i]) {
            case '"':
                if( havebackslash) {
                    // a quoted quote.  Cool
                    //
                    *(sp++) = a[i];
                    *sp = 0;
                    havebackslash = 0;
                } else {
                    // Toggle the flag
                    inquote = 1 - inquote;
                }
                break;

            case '\\':
                if( havebackslash) {
                    *(sp++) = a[i];
                    *sp = 0;
                    havebackslash = 0;
                } else {
                    havebackslash = 1;
                }
                break;

            case ',':

```

```

    if( inquote || havebackslash) {
        *(sp++) = a[i];
        *sp = 0;
        havebackslash = 0;
    } else {
        rtn[rtni++] = strdup( acums);
        sp = acums;
    }
    break;

case '}' :
    if( inquote || havebackslash) {
        *(sp++) = a[i];
        *sp = 0;
        havebackslash = 0;
    } else {
        rtn[rtni++] = strdup( acums);
        rtn[rtni] = NULL;
        free( acums);
        return( rtn);
    }
    break;

default:
    *(sp++) = a[i];
    *sp = 0;
    havebackslash = 0;
}
}
//
// Getting here means the final '}' was missing
// Probably we should throw an error or log it or something.
// Through out the last entry since this there is not resonable expectation
// that
// we should be parsing it anyway.
//
rtn[rtni] = NULL;
free( acums);
return( rtn);
}

```

7.5.4.2 void lspg_cmd_cb (lspg_query_queue_t * qqp, PGresult * pgr)

Send strings directly to PMAC queue.

Parameters

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

Definition at line 1238 of file lspg.c.

```

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

for( i=0; i<PGntuples( pgr); i++) {
    sp = PQgetvalue( pgr, i, 0);
    if( sp != NULL && *sp != 0) {
        lspmac_SockSendDPLine( NULL, sp);
        //      lspmac_SockSendline( sp);
        //
        // Keep asking for more until
        // there are no commands left
        //
        // This should solve a potential problem where
        // more than one command is put on the queue for a given notify.
        //
        lspg_query_push( lspg_cmd_cb, "select
            pmac.md2_queue_next()");
    }
}
}

```

7.5.4.3 void lspg_demandairrights_all ()

do nothing until we get airrights

Definition at line 556 of file lspg.c.

```

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

```

7.5.4.4 void lspg_demandairrights_call ()

call for airrights

Definition at line 538 of file lspg.c.

```

{
    pthread_mutex_lock( &lspg_demandairrights.mutex);
    lspg_demandairrights.new_value_ready = 0;
    pthread_mutex_unlock( &lspg_demandairrights.mutex);
    lspg_query_push( lspg_demandairrights_cb
        , "SELECT px.demandairrights()");
}

```

7.5.4.5 void lspg_demandairrights_cb (lspg_query_queue_t * qqp, PGresult * pgr)

handle the airrights response

Definition at line 529 of file lspg.c.

```

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

```

7.5.4.6 void lspg_demandairrights_init ()

initialize the demandairrights structure

Definition at line 521 of file lspg.c.

```

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

```

7.5.4.7 void lspg_demandairrights_wait ()

wait for the air rights request to return

Definition at line 547 of file lspg.c.

```

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

```


7.5.4.8 void lspg_flush ()

Flush psql output buffer (ie, send the query)

Definition at line 1268 of file lspg.c.

```

    {
        int err;

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

                lslogging_log_message( "flush failed: %s",
                    PQerrorMessage( q ));

                ls_pg_state = LS_PG_STATE_IDLE;
                //
                // We should probably reset the connection and start from scratch.
                // Probably the connection died.
                //
                break;

            case 0:
                // goodness and joy.
                ls_pg_state = LS_PG_STATE_RECV;
                break;

            case 1:
                // more sending to do
                ls_pg_state = LS_PG_STATE_SEND_FLUSH;
                break;
        }
    }
}

```

7.5.4.9 void lspg_getcenter_all ()

Convenience function to complete synchronous getcenter query.

Definition at line 1201 of file lspg.c.

```

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

```

7.5.4.10 void lspg_getcenter_call ()

Request a getcenter query.

Definition at line 1177 of file lspg.c.

```

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

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

```

7.5.4.11 void lspg_getcenter_cb (lspg_query_queue_t * qq, PGresult * pgr)

Retrieve the data to center the crystal.

Definition at line 1112 of file lspg.c.

```

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

pthread_mutex_lock( &(lspg_getcenter.mutex));

lspg_getcenter.no_rows_returned = PQntuples(
    pgr) <= 0;
if( lspg_getcenter.no_rows_returned) {
    //
    // No particular reason this path should ever be taken
    // but if we don't get rows then we had better not move anything.
    //
    lspg_getcenter.new_value_ready = 1;
    pthread_cond_signal( &(lspg_getcenter.cond));
    pthread_mutex_unlock( &(lspg_getcenter.mutex));
    return;
}

zoom_c = PQfnumber( pgr, "zoom");
dcx_c = PQfnumber( pgr, "dcx");
dcy_c = PQfnumber( pgr, "dcy");
dax_c = PQfnumber( pgr, "dax");
day_c = PQfnumber( pgr, "day");
daz_c = PQfnumber( pgr, "daz");

lspg_getcenter.zoom_isnull = PQgetisnull( pgr, 0,
    zoom_c);
if( lspg_getcenter.zoom_isnull == 0)
    lspg_getcenter.zoom = atoi( PQgetvalue( pgr, 0, zoom_c));

lspg_getcenter.dcx_isnull = PQgetisnull( pgr, 0,
    dcx_c);
if( lspg_getcenter.dcx_isnull == 0)
    lspg_getcenter.dcx = atof( PQgetvalue( pgr, 0, dcx_c));

lspg_getcenter.dcy_isnull = PQgetisnull( pgr, 0,
    dcy_c);
if( lspg_getcenter.dcy_isnull == 0)
    lspg_getcenter.dcy = atof( PQgetvalue( pgr, 0, dcy_c));

lspg_getcenter.dax_isnull = PQgetisnull( pgr, 0,
    dax_c);
if( lspg_getcenter.dax_isnull == 0)
    lspg_getcenter.dax = atof( PQgetvalue( pgr, 0, dax_c));

lspg_getcenter.day_isnull = PQgetisnull( pgr, 0,
    day_c);
if( lspg_getcenter.day_isnull == 0)
    lspg_getcenter.day = atof( PQgetvalue( pgr, 0, day_c));

lspg_getcenter.daz_isnull = PQgetisnull( pgr, 0,
    daz_c);
if( lspg_getcenter.daz_isnull == 0)
    lspg_getcenter.daz = atof( PQgetvalue( pgr, 0, daz_c));

lspg_getcenter.new_value_ready = 1;

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

```

7.5.4.12 void lspg_getcenter_done ()

Done with getcenter query.

Definition at line 1195 of file lspg.c.

```

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

```

7.5.4.13 void lspg_getcenter_init ()

Initialize getcenter object.

Definition at line 1169 of file lspg.c.

```

    {
        memset( &lspg_getcenter, 0, sizeof( lspg_getcenter
        ));
        pthread_mutex_init( &(lspg_getcenter.mutex), NULL);
        pthread_cond_init( &(lspg_getcenter.cond), NULL);
    }

```

7.5.4.14 void lspg_getcenter_wait ()

Wait for a getcenter query to return.

Definition at line 1187 of file lspg.c.

```

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

```

7.5.4.15 void lspg_getcurrentsampleid_call ()

Definition at line 367 of file lspg.c.

```

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

        lspg_query_push( lspg_getcurrentsampleid_cb
        , "SELECT px.getcurrentsampleid()");
    }

```

7.5.4.16 void lspg_getcurrentsampleid_cb (lspg_query_queue_t * qqp, PGresult * pgr)

get currentsampleid

Definition at line 346 of file lspg.c.

```

    {
        pthread_mutex_lock( &lspg_getcurrentsampleid.mutex
        );

        lspg_nextsample.new_value_ready = 1;
        lspg_getcurrentsampleid.no_rows_returned
        = PQntuples( pgr) <= 0;
        if( lspg_getcurrentsampleid.no_rows_returned
        ) {
            pthread_cond_signal( &lspg_getcurrentsampleid.cond
            );
            pthread_mutex_unlock( &lspg_getcurrentsampleid.mutex
            );
            return;
        }

        lspg_getcurrentsampleid.getcurrentsampleid_isnull
        = PQgetisnull( pgr, 0, 0);
        if( lspg_getcurrentsampleid.getcurrentsampleid_isnull
        == 0)
            lspg_getcurrentsampleid.getcurrentsampleid
            = strtol( PQgetvalue( pgr, 0, 0), NULL, 0);

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

```

7.5.4.17 void lspg_getcurrentsampleid_init ()

Definition at line 338 of file lspg.c.

```

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

```

7.5.4.18 unsigned int lspg_getcurrentsampleid_read ()

Definition at line 377 of file lspg.c.

```

{
    unsigned int rtn;
    pthread_mutex_lock( &lspg_getcurrentsampleid.mutex
        );
    while( lspg_getcurrentsampleid.new_value_ready
        == 0)
        pthread_cond_wait( &lspg_getcurrentsampleid.cond
            , &lspg_getcurrentsampleid.mutex);

    if( lspg_getcurrentsampleid.getcurrentsampleid_isnull
        )
        rtn = -1;
    else
        rtn = lspg_getcurrentsampleid.getcurrentsampleid
            ;
    pthread_mutex_unlock( &lspg_getcurrentsampleid.mutex
        );
    return rtn;
}

```

7.5.4.19 void lspg_getcurrentsampleid_wait_for_id (unsigned int test)

Definition at line 393 of file lspg.c.

```

{
    pthread_mutex_lock( &lspg_getcurrentsampleid.mutex
        );
    while( lspg_getcurrentsampleid.getcurrentsampleid
        != test)
        pthread_cond_wait( &lspg_getcurrentsampleid.cond
            , &lspg_getcurrentsampleid.mutex);

    pthread_mutex_unlock( &lspg_getcurrentsampleid.mutex
        );
}

```

7.5.4.20 void lspg_init ()

Initialize the lspg module.

Definition at line 1758 of file lspg.c.

```

{
    pthread_mutex_init( &lspg_queue_mutex, NULL);
    pthread_cond_init( &lspg_queue_cond, NULL);

    lspg_demandairrights_init();
    lspg_getcenter_init();
    lspg_getcurrentsampleid_init();
    lspg_lock_detector_init();
    lspg_lock_diffractionmeter_init();
    lspg_nextsample_init();
}

```

```

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

```

7.5.4.21 void lspg_lock_detector_all ()

Detector lock convenience function.

Definition at line 1024 of file lspg.c.

```

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

```

7.5.4.22 void lspg_lock_detector_call ()

Request (demand) a detector lock.

Definition at line 1000 of file lspg.c.

```

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

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

```

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

Callback for when the detector lock has be grabbed.

Definition at line 991 of file lspg.c.

```

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

```

7.5.4.24 void lspg_lock_detector_done ()

Finish waiting.

Definition at line 1018 of file lspg.c.

```

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

```

7.5.4.25 void lspg_lock_detector_init ()

Initialize detector lock object.

Definition at line 983 of file lspg.c.

```

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

```

7.5.4.26 void lspg_lock_detector_wait ()

Wait for the detector lock.

Definition at line 1010 of file lspg.c.

```

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

```

7.5.4.27 void lspg_lock_diffractionmeter_all ()

Convenience function that combines lock diffractionmeter calls.

Definition at line 965 of file lspg.c.

```

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

```

7.5.4.28 void lspg_lock_diffractionmeter_call ()

Request that the database grab the diffractionmeter lock.

Definition at line 941 of file lspg.c.

```

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

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

```

7.5.4.29 void lspg_lock_diffractionmeter.cb (lspg_query_queue_t * qqp, PGresult * pgr)

Callback routine for a lock diffractionmeter query.

Definition at line 932 of file lspg.c.

```

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

```

7.5.4.30 void lspg_lock_diffractionmeter_done ()

Finish up the lock diffractionmeter call.

Definition at line 959 of file lspg.c.

```

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

```

7.5.4.31 void lspg_lock_diffractionmeter_init ()

initialize the diffractionmeter locking object

Definition at line 924 of file lspg.c.

```

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

```

7.5.4.32 void lspg_lock_diffractionmeter_wait ()

Wait for the diffractionmeter lock.

Definition at line 951 of file lspg.c.

```

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

```

7.5.4.33 void lspg_next_state ()

Implements our state machine Does not strictly only set the next state as it also calls some functions that, perhaps, alters the state mid-function.

Definition at line 1623 of file lspg.c.

```

{
//
// connect to the database
//
if( q == NULL ||
ls_pg_state == LS_PG_STATE_INIT ||

```

```

    ls_pg_state == LS_PG_STATE_RESET ||
    ls_pg_state == LS_PG_STATE_INIT_POLL ||
    ls_pg_state == LS_PG_STATE_RESET_POLL)
    lspg_pg_connect( lspgfd);

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

switch( ls_pg_state) {
case LS_PG_STATE_INIT_POLL:
    if( lspg_connectPoll_response ==
        PGRES_POLLING_WRITING)
        lspgfd.events = POLLOUT;
    else if( lspg_connectPoll_response ==
        PGRES_POLLING_READING)
        lspgfd.events = POLLIN;
    else
        lspgfd.events = 0;
    break;

case LS_PG_STATE_RESET_POLL:
    if( lspg_resetPoll_response == PGRES_POLLING_WRITING
        )
        lspgfd.events = POLLOUT;
    else if( lspg_resetPoll_response ==
        PGRES_POLLING_READING)
        lspgfd.events = POLLIN;
    else
        lspgfd.events = 0;
    break;

case LS_PG_STATE_IDLE:
case LS_PG_STATE_RECV:
    lspgfd.events = POLLIN;
    break;

case LS_PG_STATE_SEND:
case LS_PG_STATE_SEND_FLUSH:
    lspgfd.events = POLLOUT;
    break;

default:
    lspgfd.events = 0;
}
}

```

7.5.4.34 void lspg_nextaction.cb (lspg_query_queue_t * qqp, PGresult * pgr)

Queue the next MD2 instruction.

Parameters

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

Definition at line 1210 of file lspg.c.

```

{
char *action;

if( PQntuples( pgr) <= 0)
    return; // Note: nextaction should always return at least
            "noAction", so this branch should never be taken

action = PQgetvalue( pgr, 0, 0); // next action only returns one row

if( strcmp( action, "noAction") == 0)
    return;

if( pthread_mutex_trylock( &md2cmds_mutex) == 0) {
    strncpy( md2cmds_cmd, action, MD2CMDS_CMD_LENGTH
            -1);
    md2cmds_cmd[MD2CMDS_CMD_LENGTH-1] = 0;
    pthread_cond_signal( &md2cmds_cond);
    pthread_mutex_unlock( &md2cmds_mutex);
} else {

```



```

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

```

7.5.4.35 unsigned int lspg_nextsample_all(int * err)

Definition at line 468 of file lspg.c.

```

{
    unsigned int rtn;

    lspg_nextsample_call();
    lspg_nextsample_wait();

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

    return rtn;
}

```

7.5.4.36 void lspg_nextsample_call()

Queue up a nextsample query.

Definition at line 445 of file lspg.c.

```

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

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

```

7.5.4.37 void lspg_nextsample_cb(lspg_query_queue_t * qq, PGresult * pgr)

Next Sample.

Parameters

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

Definition at line 404 of file lspg.c.

```

{
    static int got_columns = 0;
    static int nextsample_col;
    pthread_mutex_lock( &(lspg_nextsample.mutex));

    lspg_nextsample.no_rows_returned = PQntuples(
        pgr) <= 0;
    if( lspg_nextsample.no_rows_returned) {
        lslogging_log_message( "lspg_nextsample_cb: no rows
            returned. This should never happen.");
    }
}

```

```

    lspg_nextsample.new_value_ready = 1;
    pthread_cond_signal( &(lspg_nextsample.cond));
    pthread_mutex_unlock( &(lspg_nextsample.mutex));
    return;
}

if( got_columns == 0) {
    nextsample_col = PQfnumber( pgr, "nextsample");
    got_columns = 1;
}

lspg_nextsample.nextsample_isnull =
    PQgetisnull( pgr, 0, nextsample_col);
if( lspg_nextsample.nextsample_isnull == 0)
    lspg_nextsample.nextsample = strtol( PQgetvalue(
        pgr, 0, nextsample_col), NULL, 0);

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

```

7.5.4.38 void lspg_nextsample.done ()

Called when the next shot query has been processed.

Definition at line 463 of file lspg.c.

```

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

```

7.5.4.39 void lspg_nextsample.init ()

Initialize the nextsample variable, mutex, and condition.

Definition at line 437 of file lspg.c.

```

    {
        memset( &lspg_nextsample, 0, sizeof( lspg_nextsample
        ));
        pthread_mutex_init( &(lspg_nextsample.mutex), NULL);
        pthread_cond_init( &(lspg_nextsample.cond), NULL);
    }

```

7.5.4.40 void lspg_nextsample.wait ()

Wait for the nextsample query to get processed.

Definition at line 455 of file lspg.c.

```

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

```

7.5.4.41 void lspg_nextshot.call ()

Queue up a nextshot query.

Definition at line 824 of file lspg.c.

```

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

lspg_query_push( lspg_nextshot_cb, "SELECT *
    FROM px.nextshot2()");
}

```

7.5.4.42 void lspg_nextshot.cb (lspg_query_queue_t * qqp, PGresult * pgr)

Next Shot Callback.

This is a long and tedious routine as there are a large number of variables returned. Suck it up. Return with the global object lspg_nextshot set.

Parameters

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

Definition at line 569 of file lspg.c.

```

    {
static int got_col_nums=0;
static int
    dsdir_c, dspid_c, dsowidth_c, dsoscaxis_c, dsexp_c, skey_c, sstart_c, sfn_c
    , dsphi_c,
    dsomega_c, dskappa_c, dsdist_c, dsnrng_c, dshpid_c, cx_c, cy_c, ax_c, ay_c,
    az_c,
    active_c, sindex_c, stype_c,
    dsowidth2_c, dsoscaxis2_c, dsexp2_c, sstart2_c, dsphi2_c, dsomega2_c,
    dskappa2_c, dsdist2_c, dsnrng2_c,
    cx2_c, cy2_c, ax2_c, ay2_c, az2_c, active2_c, sindex2_c, stype2_c;

pthread_mutex_lock( &(lspg_nextshot.mutex));

lspg_nextshot.no_rows_returned = PQntuples( pgr)
    <= 0;
if( lspg_nextshot.no_rows_returned) {
    lspg_nextshot.new_value_ready = 1;
    pthread_cond_signal( &(lspg_nextshot.cond));
    pthread_mutex_unlock( &(lspg_nextshot.mutex));
    return; // I guess there was no shot after all
}

if( got_col_nums == 0) {
    dsdir_c = PQfnumber( pgr, "dsdir");
    dspid_c = PQfnumber( pgr, "dspid");
    dsowidth_c = PQfnumber( pgr, "dsowidth");
    dsoscaxis_c = PQfnumber( pgr, "dsoscaxis");
    dsexp_c = PQfnumber( pgr, "dsexp");
    skey_c = PQfnumber( pgr, "skey");
    sstart_c = PQfnumber( pgr, "sstart");
    sfn_c = PQfnumber( pgr, "sfn");
    dsphi_c = PQfnumber( pgr, "dsphi");
    dsomega_c = PQfnumber( pgr, "dsomega");
    dskappa_c = PQfnumber( pgr, "dskappa");
    dsdist_c = PQfnumber( pgr, "dsdist");
    dsnrng_c = PQfnumber( pgr, "dsnrng");
    dshpid_c = PQfnumber( pgr, "dshpid");
    cx_c = PQfnumber( pgr, "cx");
    cy_c = PQfnumber( pgr, "cy");
    ax_c = PQfnumber( pgr, "ax");
    ay_c = PQfnumber( pgr, "ay");
    az_c = PQfnumber( pgr, "az");
    active_c = PQfnumber( pgr, "active");
    sindex_c = PQfnumber( pgr, "sindex");
    stype_c = PQfnumber( pgr, "stype");
    dsowidth2_c = PQfnumber( pgr, "dsowidth2");
    dsoscaxis2_c = PQfnumber( pgr, "dsoscaxis2");
    dsexp2_c = PQfnumber( pgr, "dsexp2");
    sstart2_c = PQfnumber( pgr, "sstart2");
    dsphi2_c = PQfnumber( pgr, "dsphi2");
    dsomega2_c = PQfnumber( pgr, "dsomega2");
    dskappa2_c = PQfnumber( pgr, "dskappa2");
    dsdist2_c = PQfnumber( pgr, "dsdist2");
    dsnrng2_c = PQfnumber( pgr, "dsnrng2");
}

```

```

    cx2_c      = PQfnumber( pgr, "cx2");
    cy2_c      = PQfnumber( pgr, "cy2");
    ax2_c      = PQfnumber( pgr, "ax2");
    ay2_c      = PQfnumber( pgr, "ay2");
    az2_c      = PQfnumber( pgr, "az2");
    active2_c   = PQfnumber( pgr, "active2");
    sindex2_c   = PQfnumber( pgr, "sindex2");
    stype2_c    = PQfnumber( pgr, "stype2");

    got_col_nums = 1;
}

//
// NULL string values come back as empty strings
// Mark the null flag but allocate the empty string anyway
//

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

lspg_nextshot.dspid_isnull = PQgetisnull( pgr, 0,
    dspid_c);
if( lspg_nextshot.dspid != NULL)
    free( lspg_nextshot.dspid);
lspg_nextshot.dspid = strdup( PQgetvalue( pgr, 0, dspid_c))
;

lspg_nextshot.dsoscaxis_isnull = PQgetisnull(
    pgr, 0, dsoscaxis_c);
if( lspg_nextshot.dsoscaxis != NULL)
    free( lspg_nextshot.dsoscaxis);
lspg_nextshot.dsoscaxis = strdup( PQgetvalue( pgr, 0,
    dsoscaxis_c));

lspg_nextshot.dsoscaxis2_isnull = PQgetisnull(
    pgr, 0, dsoscaxis2_c);
if( lspg_nextshot.dsoscaxis2 != NULL)
    free( lspg_nextshot.dsoscaxis2);
lspg_nextshot.dsoscaxis2 = strdup( PQgetvalue( pgr, 0,
    dsoscaxis2_c));

lspg_nextshot.sfn_isnull = PQgetisnull(pgr, 0, sfn_c);
if( lspg_nextshot.sfn != NULL)
    free( lspg_nextshot.sfn);
lspg_nextshot.sfn = strdup( PQgetvalue( pgr, 0, sfn_c));

lspg_nextshot.stype_isnull = PQgetisnull( pgr, 0,
    stype_c);
if( lspg_nextshot.stype != NULL)
    free( lspg_nextshot.stype);
lspg_nextshot.stype = strdup( PQgetvalue( pgr, 0, stype_c))
;

lspg_nextshot.stype2_isnull = PQgetisnull( pgr, 0,
    stype2_c);
if( lspg_nextshot.stype2 != NULL)
    free( lspg_nextshot.stype2);
lspg_nextshot.stype2 = strdup( PQgetvalue( pgr, 0,
    stype2_c));

//
// Probably shouldn't try to convert null number values
//
lspg_nextshot.dsowidth_isnull = PQgetisnull( pgr,
    0, dsowidth_c);
if( lspg_nextshot.dsowidth_isnull == 0)
    lspg_nextshot.dsowidth = atof( PQgetvalue( pgr,0,
    dsowidth_c));

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

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

lspg_nextshot.dsphi_isnull = PQgetisnull( pgr, 0,

```

```

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```

```

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

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

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

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

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

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

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

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

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

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

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

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

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

lspg_nextshot.new_value_ready = 1;

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

```

7.5.4.43 void lspg_nextshot.done ()

Called when the next shot query has been processed.

Definition at line 842 of file lspg.c.

```

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

```

7.5.4.44 void lspg_nextshot_init ()

Initialize the nextshot variable, mutex, and condition.

Definition at line 816 of file lspg.c.

```

    {
        memset( &lspg_nextshot, 0, sizeof( lspg_nextshot));
        pthread_mutex_init( &(lspg_nextshot.mutex), NULL);
        pthread_cond_init( &(lspg_nextshot.cond), NULL);
    }

```

7.5.4.45 void lspg_nextshot_wait ()

Wait for the next shot query to get processed.

Definition at line 834 of file lspg.c.

```

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

```

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

Definition at line 1527 of file lspg.c.

```

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

```

7.5.4.47 void lspg_pg_connect ()

Connect to the pg server.

Definition at line 1534 of file lspg.c.

```

    {
        int err;

        if( q == NULL)
            ls_pg_state = LS_PG_STATE_INIT;

        switch( ls_pg_state) {
        case LS_PG_STATE_INIT:

            if( lspg_time_sent.tv_sec != 0) {
                //
                // Reality check: if it's less the about 10 seconds since the last failed
                // attempt
                // the just chill.
                //
                gettimeofday( &now, NULL);
                if( now.tv_sec - lspg_time_sent.tv_sec < 10) {
                    return;
                }
            }

            q = PQconnectStart( "dbname=ls user=lsuser hostaddr=10.1.0.3");

```

```

if( q == NULL) {
    lslogging_log_message( "Out of memory
        (lspg_pg_connect)");
    exit( -1);
}

err = PQstatus( q);
if( err == CONNECTION_BAD) {
    lslogging_log_message( "Trouble connecting to
        database");

    gettimeofday( &lspg_time_sent, NULL);
    return;
}
err = PQsetnonblocking( q, 1);
if( err != 0) {
    lslogging_log_message( "Odd, could not set database
        connection to nonblocking");
}

ls_pg_state = LS_PG_STATE_INIT_POLL;
lspg_connectPoll_response = PGRES_POLLING_WRITING;
//
// set up the connection for poll
//
lspgfd.fd = PQsocket( q);
break;

case LS_PG_STATE_INIT_POLL:
    if( lspg_connectPoll_response ==
        PGRES_POLLING_FAILED) {
        PQfinish( q);
        q = NULL;
        ls_pg_state = LS_PG_STATE_INIT;
    } else if( lspg_connectPoll_response ==
        PGRES_POLLING_OK) {
        PQsetNoticeProcessor( q, (PQnoticeProcessor)lspg_notice_processor
            , NULL);
        lspg_query_push( NULL, "select pmac.md2_init()");
        ls_pg_state = LS_PG_STATE_IDLE;
    }
    break;

case LS_PG_STATE_RESET:
    err = PQresetStart( q);
    if( err == 0) {
        PQfinish( q);
        q = NULL;
        ls_pg_state = LS_PG_STATE_INIT;
    } else {
        ls_pg_state = LS_PG_STATE_RESET_POLL;
        lspg_resetPoll_response = PGRES_POLLING_WRITING;
    }
    break;

case LS_PG_STATE_RESET_POLL:
    if( lspg_resetPoll_response == PGRES_POLLING_FAILED)
    {
        PQfinish( q);
        q = NULL;
        ls_pg_state = LS_PG_STATE_INIT;
    } else if( lspg_resetPoll_response ==
        PGRES_POLLING_OK) {
        lspg_query_push( NULL, "select pmac.md2_init()");
        ls_pg_state = LS_PG_STATE_IDLE;
    }
    break;
}
}

```

7.5.4.48 void lspg_pg_service (struct pollfd * evt)

I/O control to/from the postgresql server.

Parameters

in	evt	The pollfd object that we are responding to
----	-----	---

Definition at line 1428 of file lspg.c.


```

        {
//
// Currently just used to check for notifies
// Other socket communication is done synchronously
//

if( evt->revents & POLLIN) {
    int err;

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

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

//
// if in IDLE or RECV we need to call consumeInput first
//
if( ls_pg_state == LS_PG_STATE_IDLE) {
    err = PQconsumeInput( q);
    if( err != 1) {
        lslogging_log_message( "consume input failed: %s",
            PQerrorMessage( q));
        ls_pg_state = LS_PG_STATE_RESET;
        return;
    }
}

if( ls_pg_state == LS_PG_STATE_RECV) {
    lspg_receive();
}

//
// Check for notifies regardless of our state
// Push as many requests as we have notifies.
//
{
    PGnotify *pgn;

    while( 1) {
        pgn = PQnotifies( q);
        if( pgn == NULL)
            break;

        lslogging_log_message( "lspg_pg_service: notify
            recieved %s", pgn->relnam);

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

if( evt->revents & POLLOUT) {

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

```

```

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

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

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

```

7.5.4.49 lspg_query_queue_t* lspg_query_next()

Return the next item in the postgresql queue.

If there is an item left in the queue then it is returned. Otherwise, NULL is returned.

Definition at line 75 of file lspg.c.

```

{
    lspg_query_queue_t *rtn;

    pthread_mutex_lock( &lspg_queue_mutex);

    if( lspg_query_queue_off == lspg_query_queue_on
    )
        // Queue is empty
        rtn = NULL;
    else {
        rtn = &(lspg_query_queue[ (lspg_query_queue_off
        ++ ) % LS_PG_QUERY_QUEUE_LENGTH]);
        pthread_cond_signal( &lspg_queue_cond);
    }
    pthread_mutex_unlock( &lspg_queue_mutex);

    return rtn;
}

```

7.5.4.50 void lspg_query_push(void(*) (lspg_query_queue_t *, PGresult *) cb, char * fmt, ...)

Place a query on the queue.

Parameters

in	<i>cb</i>	Our callback function that deals with the response
in	<i>fmt</i>	Printf style function to generate the query

Definition at line 128 of file lspg.c.

```

{
    int idx;
    va_list arg_ptr;

    pthread_mutex_lock( &lspg_queue_mutex);

    //
    // Pause the thread while we service the queue
    //
    while( (lspg_query_queue_on + 1) %
        LS_PG_QUERY_QUEUE_LENGTH == lspg_query_queue_off %
        LS_PG_QUERY_QUEUE_LENGTH) {
        pthread_cond_wait( &lspg_queue_cond, &lspg_queue_mutex

```

```

    );
}

idx = lspg_query_queue_on % LS_PG_QUERY_QUEUE_LENGTH
;

va_start( arg_ptr, fmt);
vsnprintf( lspg_query_queue[idx].qs,
    LS_PG_QUERY_STRING_LENGTH-1, fmt, arg_ptr);
va_end( arg_ptr);

lspg_query_queue[idx].qs[LS_PG_QUERY_STRING_LENGTH
    - 1] = 0;
lspg_query_queue[idx].onResponse = cb;
lspg_query_queue_on++;

pthread_kill( lspg_thread, SIGUSR1);
pthread_mutex_unlock( &lspg_queue_mutex);
};

```

7.5.4.51 void lspg_query_reply_next ()

Remove the oldest item in the queue.

this is called only when there is nothing else to service the reply: this pop does not return anything. We use the ...reply_peek function to return the next item in the reply queue

Definition at line 99 of file lspg.c.

```

{
pthread_mutex_lock( &lspg_queue_mutex);

if( lspg_query_queue_reply != lspg_query_queue_on
)
    lspg_query_queue_reply++;

pthread_mutex_unlock( &lspg_queue_mutex);
}

```

7.5.4.52 lspg_query_queue_t* lspg_query_reply_peek ()

Return the next item in the reply queue but don't pop it since we may need it more than once.

Call [lspg_query_reply_next\(\)](#) when done.

Definition at line 112 of file lspg.c.

```

{
lspg_query_queue_t *rtn;

pthread_mutex_lock( &lspg_queue_mutex);

if( lspg_query_queue_reply == lspg_query_queue_on
)
    rtn = NULL;
else
    rtn = &(lspg_query_queue[ (lspg_query_queue_reply
    ) % LS_PG_QUERY_QUEUE_LENGTH]);

pthread_mutex_unlock( &lspg_queue_mutex);
return rtn;
}

```

7.5.4.53 void lspg_receive ()

Receive a result of a query.

Definition at line 1345 of file lspg.c.

```

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

    err = PQconsumeInput( q );
    if( err != 1 ) {
        lslogging_log_message( "consume input failed: %s",
            PQerrorMessage( q ));
        ls_pg_state = LS_PG_STATE_RESET;
        return;
    }

    //
    // We must call PQgetResult until it returns NULL before sending the next
    // query
    // This implies that only one query can ever be active at a time and our
    // queue
    // management should be simple
    //
    // We should be in the LS_PG_STATE_RECV here
    //

    while( !PQisBusy( q )) {
        pgr = PQgetResult( q );
        if( pgr == NULL ) {
            lspg_query_reply_next();
            //
            // we are now done reading the response from the database
            //
            ls_pg_state = LS_PG_STATE_IDLE;
            break;
        } else {
            ExecStatusType es;

            qqp = lspg_query_reply_peek();
            es = PQresultStatus( pgr);

            if( es != PGRES_COMMAND_OK && es != PGRES_TUPLES_OK ) {
                char *emess;
                emess = PQresultErrorMessage( pgr);
                if( emess != NULL && emess[0] != 0 ) {
                    lslogging_log_message( "Error from query '%s':\n
%s", qqp->qid, emess);
                }
            } else {
                //
                // Deal with the response
                //
                // If the response is likely to take awhile we should probably
                // add a new state and put something in the main look to run the
                onResponse
                // routine in the main loop. For now, though, we only expect very
                brief onResponse routines
                //
                if( qqp != NULL && qqp->onResponse != NULL )
                    qqp->onResponse( qqp, pgr);
            }
            PQclear( pgr);
        }
    }
}

```

7.5.4.54 void lspg_run ()

Start 'er runnin'.

Definition at line 1777 of file lspg.c.

```

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

```

7.5.4.55 void lspg_send_next_query ()

send the next queued query to the DB server

Definition at line 1298 of file lspg.c.

```

    {
//
// Normally we should be in the "send" state
// but we can also send if we are servicing
// a reply
//

lspg_query_queue_t *qqp;
int err;

qqp = lspg_query_next();
if( qqp == NULL) {
//
// A send without a query? Should never happen.
// But at least we shouldn't segfault if it does.
//
return;
}

if( qqp->qs[0] == 0) {
//
// Do we really have to check this case?
// It would only come up if we stupidly pushed an empty query string
// or ran off the end of the queue
//
lslogging_log_message( "Popped empty query string.
    Probably bad things are going on.");

lspg_query_reply_next();
ls_pg_state = LS_PG_STATE_IDLE;
} else {
err = PQsendQuery( q, qqp->qs);
if( err == 0) {
lslogging_log_message( "query failed: %s\n",
    PQerrorMessage( q));

//
// Don't wait for a reply, just reset the connection
//
lspg_query_reply_next();
ls_pg_state = LS_PG_STATE_RESET;
} else {
ls_pg_state = LS_PG_STATE_SEND_FLUSH;
}
}
}
}

```

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

Convenience function to call seq run prep.

Parameters

in	<i>skey</i>	px.shots key for this image
in	<i>kappa</i>	current kappa postion
in	<i>phi</i>	current phi postition
in	<i>cx</i>	current center table x
in	<i>cy</i>	current center table y
in	<i>ax</i>	current alignment table x
in	<i>ay</i>	current alignment table y
in	<i>az</i>	current alignment table z

Definition at line 1095 of file lspg.c.

```

{
lspg_seq_run_prep_call( skey, kappa, phi, cx,
    cy, ax, ay, az);
lspg_seq_run_prep_wait();
lspg_seq_run_prep_done();
}

```

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

queue up the `seq_run_prep` query

Parameters

in	<i>skey</i>	px.shots key for this image
in	<i>kappa</i>	current kappa postion
in	<i>phi</i>	current phi postition
in	<i>cx</i>	current center table x
in	<i>cy</i>	current center table y
in	<i>ax</i>	current alignment table x
in	<i>ay</i>	current alignment table y
in	<i>az</i>	current alignment table z

Definition at line 1061 of file `lspg.c`.

```

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

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

```

7.5.4.58 void `lspg_seq_run_prep_cb` (`lspg_query_queue_t` * *qqp*, `PGresult` * *pgr*)

Callback for the `seq_run_prep` query.

Parameters

in	<i>qqp</i>	The query item that generated this callback
in	<i>pgr</i>	The result of the query

Definition at line 1049 of file `lspg.c`.

```

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

```

7.5.4.59 void `lspg_seq_run_prep_done` ()

Indicate we are done waiting.

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

```

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

```

7.5.4.60 void `lspg_seq_run_prep_init` ()

Initialize the data collection object.

Definition at line 1041 of file `lspg.c`.

```

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

```

7.5.4.61 void lspg_seq_run_prep.wait ()

Wait for seq run prep query to return.

Definition at line 1081 of file lspg.c.

```

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

```

7.5.4.62 void lspg_sig.service (struct pollfd * evt)

Service a signal Signals here are treated as file descriptors and fits into our poll scheme.

Parameters

in	evt	The pollfd object that triggered this call
----	-----	--

Definition at line 1406 of file lspg.c.

```

    {
        struct signalfd_sinfo 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_sinfo));
    }

```

7.5.4.63 int lspg_starttransfer.all (int * err, unsigned int nextsample, int sampledected, double ax, double ay, double az, double horz, double vert, double esttime)

Definition at line 322 of file lspg.c.

```

    {
        int rtn;

        lspg_starttransfer_call( nextsample, sampledected,
            ax, ay, az, horz, vert, esttime);
        lspg_starttransfer_wait();
        if( lspg_starttransfer.no_rows_returned ||
            lspg_starttransfer.starttransfer != 1) {
            *err = 1;
        } else {
            *err = 0;
            rtn = lspg_starttransfer.starttransfer;
        }
        lspg_starttransfer_done();
    }

```

```

    return rtn;
}

```

7.5.4.64 void lspg_starttransfer_call (unsigned int *nextsample*, int *sample_detected*, double *ax*, double *ay*, double *az*, double *horz*, double *vert*, double *esttime*)

Definition at line 302 of file lspg.c.

```

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

    lspg_query_push( lspg_starttransfer_cb, "
        SELECT px.starttransfer( %d, %d, %.3f, %.3f, %.3f, %.3f, %.3f, %.3f",
            nextsample, sample_detected, ax, ay, az, horz
        , vert, esttime);
}

```

7.5.4.65 void lspg_starttransfer_cb (lspg_query_queue_t * *qqp*, PGresult * *pgr*)

Parameters

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

Definition at line 281 of file lspg.c.

```

{
    pthread_mutex_lock( &(lspg_starttransfer.mutex));

    lspg_starttransfer.new_value_ready = 1;
    if( PQntuples( pgr) <=0) {
        lspg_starttransfer.no_rows_returned = 0;
        lspg_starttransfer.starttransfer = 0;
    } else {
        lspg_starttransfer.no_rows_returned = 1;
        if( PQgetisnull( pgr, 0, 0) || strtol( PQgetvalue( pgr, 0, 0), NULL, 0) !=
            1)
            lspg_starttransfer.starttransfer = 0;
        else
            lspg_starttransfer.starttransfer = 1;
    }
    pthread_cond_signal( &(lspg_starttransfer.cond));
    pthread_mutex_unlock( &(lspg_starttransfer.mutex));
}

```

7.5.4.66 void lspg_starttransfer_done ()

Definition at line 317 of file lspg.c.

```

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

```

7.5.4.67 void lspg_starttransfer_init ()

Definition at line 275 of file lspg.c.

```

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

```


7.5.4.68 void lspg_starttransfer_wait ()

Definition at line 311 of file lspg.c.

```

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

```

7.5.4.69 void lspg_wait_for_detector_all ()

Combined call to wait for the detector.

Definition at line 905 of file lspg.c.

```

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

```

7.5.4.70 void lspg_wait_for_detector.call ()

initiate the wait for detector query

Definition at line 879 of file lspg.c.

```

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

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

```

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

Callback for the wait for detector query.

Definition at line 870 of file lspg.c.

```

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

```

7.5.4.72 void lspg_wait_for_detector.done ()

Done waiting for the detector.

Definition at line 898 of file lspg.c.

```

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

```

7.5.4.73 void lspg_wait_for_detector_init ()

initialize the detector timing object

Definition at line 862 of file lspg.c.

```

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

```

7.5.4.74 void lspg_wait_for_detector_wait ()

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

Definition at line 890 of file lspg.c.

```

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

```

7.5.4.75 void lspg_waitcryo_all ()

no need to get fancy with the wait cryo command It should not return until the robot is almost ready for air rights

Definition at line 507 of file lspg.c.

```

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

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

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

pthread_mutex_unlock( &lspg_waitcryo.mutex);
}

```

7.5.4.76 void lspg_waitcryo.cb (lspg_query_queue_t * qqp, PGresult * pgr)

Definition at line 497 of file lspg.c.

```

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

```

7.5.4.77 void lspg_waitcryo_init ()

Definition at line 491 of file lspg.c.

```

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

```

7.5.4.78 void* lspg_worker (void * dummy)

The main loop for the lspg thread.

Parameters

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

Definition at line 1674 of file lspg.c.

```

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

        //
        // block ordinary signal mechanism
        //
        sigemptyset( &our_sigset);
        sigaddset( &our_sigset, SIGUSR1);
        pthread_sigmask(SIG_BLOCK, &our_sigset, NULL);

        fda[0].fd = signalfd( -1, &our_sigset, SFD_NONBLOCK);
        if( fda[0].fd == -1) {
            char *es;

            es = strerror( errno);
            lslogging_log_message( "Signalfd trouble: %s", es);
        }
        fda[0].events = POLLIN;

        //
        // make sure file descriptor is not legal until it's been conneceted
        //
        lspgfd.fd = -1;

        while( 1) {
            int pollrtn;
            int poll_timeout_ms;

            lspg_next_state();

            if( lspgfd.fd == -1) {
                //
                // Here a connection to the database is not established.
                // Periodicaly try again. Should possibly arrange to reconnect
                // to signalfd but that's unlikely to be nessesary.
                //
                nfda = 1;
                poll_timeout_ms = 10000;
                fda[1].revents = 0;
            } else {
                //
                // Arrange to peacefully do nothing until either the pg server sends us
                // something
                // or someone pushes something onto our queue
                //
                nfda = 2;
                fda[1].fd = lspgfd.fd;
                fda[1].events = lspgfd.events;
                fda[1].revents = 0;
                poll_timeout_ms = -1;
            }

            pollrtn = poll( fda, nfda, poll_timeout_ms);

```

```

    if( pollrtn && fda[0].revents) {
        lspg_sig_service( &(fda[0]));
        pollrtn--;
    }
    if( pollrtn && fda[1].revents) {
        lspg_pg_service( &(fda[1]));
        pollrtn--;
    }
}
}

```

7.5.4.79 void lspmac_sample_detector_cb (char * event)

log magnet state

Definition at line 1746 of file lspg.c.

```

{
    int present;
    if( strcmp( event, "SampleDetected") == 0)
        present = 1;
    else
        present = 0;

    lspg_query_push( NULL, "SELECT px.logmagnetstate(%s)", present
        ? "TRUE" : "FALSE");
}

```

7.5.5 Variable Documentation

7.5.5.1 int ls_pg_state = LS_PG_STATE_INIT [static]

State of the lspg state machine.

Definition at line 39 of file lspg.c.

7.5.5.2 PostgresPollingStatusType lspg_connectPoll_response [static]

Used to determine state while connecting.

Definition at line 60 of file lspg.c.

7.5.5.3 lspg_demandairrights_t lspg_demandairrights

our demandairrights object

Definition at line 66 of file lspg.c.

7.5.5.4 lspg_getcenter_t lspg_getcenter

the getcenter object

Definition at line 65 of file lspg.c.

7.5.5.5 lspg_getcurrentsampleid_t lspg_getcurrentsampleid

our currentsample id

Definition at line 67 of file lspg.c.

7.5.5.6 `lspg_lock_detector_t lspg_lock_detector` `[static]`

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

7.5.5.7 `lspg_lock_diffractionmeter_t lspg_lock_diffractionmeter` `[static]`

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

7.5.5.8 `lspg_nextsample_t lspg_nextsample`

the very next sample

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

7.5.5.9 `lspg_nextshot_t lspg_nextshot`

the nextshot object

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

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

Our query queue.

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

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

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

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

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

Next position to add something to the queue.

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

7.5.5.13 `unsigned int lspg_query_queue_reply = 0` `[static]`

The current item being digested.

Normally `off <= reply <= on`. Corner case of queue wrap around works because we only increment and compare for equality.

Definition at line 55 of file `lspg.c`.

7.5.5.14 `pthread_cond_t lspg_queue_cond` `[static]`

keeps the queue from overflowing

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

7.5.5.15 pthread_mutex_t lspg_queue_mutex [static]

keep the queue from getting tangled

Definition at line 43 of file lspg.c.

7.5.5.16 PostgresPollingStatusType lspg_resetPoll_response [static]

Used to determine state while reconnecting.

Definition at line 61 of file lspg.c.

7.5.5.17 lspg_seq_run_prep_t lspg_seq_run_prep [static]

Definition at line 1037 of file lspg.c.

7.5.5.18 lspg_starttransfer_t lspg_starttransfer

start a sample transfer

Definition at line 68 of file lspg.c.

7.5.5.19 pthread_t lspg_thread [static]

our worker thread

Definition at line 42 of file lspg.c.

7.5.5.20 lspg_wait_for_detector_t lspg_wait_for_detector [static]

Instance of the detector timing object.

Definition at line 858 of file lspg.c.

7.5.5.21 lspg_waitcryo_t lspg_waitcryo

signal the robot

Definition at line 69 of file lspg.c.

7.5.5.22 struct pollfd lspgfd [static]

our poll info

Definition at line 45 of file lspg.c.

7.5.5.23 struct timeval lspg_time_sent now [static]

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

Definition at line 40 of file lspg.c.

7.5.5.24 PGconn* q = NULL [static]

Database connector.

Definition at line 59 of file lspg.c.

7.6 lspmac.c File Reference

Routines concerned with communication with PMAC.

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

Data Structures

- struct [md2StatusStruct](#)
The block of memory retrieved in a status request.
- struct [lspmac_ascii_buffers_struct](#)
- struct [lspmac_dpascii_queue_struct](#)

Macros

- #define [LS_PMAC_STATE_RESET](#) -1
- #define [LS_PMAC_STATE_DETACHED](#) 0
- #define [LS_PMAC_STATE_IDLE](#) 1
- #define [LS_PMAC_STATE_SC](#) 2
- #define [LS_PMAC_STATE_WACK_NFR](#) 3
- #define [LS_PMAC_STATE_WACK_CC](#) 4
- #define [LS_PMAC_STATE_WACK](#) 5
- #define [LS_PMAC_STATE_GMR](#) 6
- #define [LS_PMAC_STATE_CR](#) 7
- #define [LS_PMAC_STATE_RR](#) 8
- #define [LS_PMAC_STATE_WACK_RR](#) 9
- #define [LS_PMAC_STATE_GB](#) 10
- #define [LS_PMAC_STATE_WCR](#) 11
- #define [LS_PMAC_STATE_WGB](#) 12
- #define [LSPMAC_PRESET_REGEX](#) "(.*\\.%s\\.presets)\\.[0-9]+\\.(name|position)"
Regex to pick out preset name and corresponding position.
- #define [PMACPORT](#) 1025
The PMAC (only) listens on this port.
- #define [pmac_cmd_size](#) 8
PMAC command size in bytes.
- #define [VR_UPLOAD](#) 0xc0
- #define [VR_DOWNLOAD](#) 0x40
- #define [VR_PMAC_SENDLINE](#) 0xb0
- #define [VR_PMAC_GETLINE](#) 0xb1
- #define [VR_PMAC_FLUSH](#) 0xb3
- #define [VR_PMAC_GETMEM](#) 0xb4
- #define [VR_PMAC_SETMEM](#) 0xb5
- #define [VR_PMAC_SENDCTRLCHAR](#) 0xb6
- #define [VR_PMAC_SETBIT](#) 0xba
- #define [VR_PMAC_SETBITS](#) 0xbb
- #define [VR_PMAC_PORT](#) 0xbe

- `#define VR_PMAC_GETRESPONSE 0xbf`
- `#define VR_PMAC_READREADY 0xc2`
- `#define VR_CTRL_RESPONSE 0xc4`
- `#define VR_PMAC_GETBUFFER 0xc5`
- `#define VR_PMAC_WRITEBUFFER 0xc6`
- `#define VR_PMAC_WRITEERROR 0xc7`
- `#define VR_FWDOWNLOAD 0xcb`
- `#define VR_IPADDRESS 0xe0`
- `#define PMAC_MIN_CMD_TIME 10000.0`
Minimum time between commands to the pmac.
- `#define PMAC_CMD_QUEUE_LENGTH 2048`
Size of the PMAC command queue.
- `#define LSPMAC_DPASCII_QUEUE_LENGTH 1024`

Typedefs

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

Functions

- `void lspmac_get_ascii (char *)`
Forward declarateion.
- `double lspmac_lut (int nlut, double *lut, double x)`
Look up table support for motor positions (think x=zoom, y=light intensity) use a lookup table to find the "counts" to move the motor to the requested position The look up table is a simple one dimensional array with the x values as even indicies and the y values as odd indices.
- `double lspmac_rlut (int nlut, double *lut, double y)`
- `void hex_dump (int n, unsigned char *s)`
Prints a hex dump of the given data.
- `void cleanstr (char *s)`
Replace \r with \n in null terminated string and print result to terminal.
- `void lsConnect (char *ipaddr)`
Connect to the PMAC socket.
- `void lspmac_reset_queue ()`
Clear the queue as part of PMAC reinitialization.
- `pmac_cmd_queue_t * lspmac_push_queue (pmac_cmd_queue_t *cmd)`
Put a new command on the queue.
- `pmac_cmd_queue_t * lspmac_pop_queue ()`
Remove the oldest queue item.
- `pmac_cmd_queue_t * lspmac_pop_reply ()`
Remove the next command queue item that is waiting for a reply.
- `pmac_cmd_queue_t * lspmac_send_command (int rqType, int rq, int wValue, int wIndex, int wLength, char *data, void(*responseCB)(pmac_cmd_queue_t *, int, char *), int no_reply, char *event)`
Compose a packet and send it to the PMAC.
- `void lspmac_SockFlush ()`
Reset the PMAC socket from the PMAC side.
- `void lspmac_Reset ()`

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

 - void [lspmac_Error](#) (char *buff)
- The service routing detected an error condition.*

 - void [lspmac_Service](#) (struct pollfd *evt)
- Service routine for packet coming from the PMAC.*

 - void [lspmac_GetShortReplyCB](#) (pmac_cmd_queue_t *cmd, int nreceived, char *buff)
- Receive a reply that does not require multiple buffers.*

 - void [lspmac_SendControlReplyPrintCB](#) (pmac_cmd_queue_t *cmd, int nreceived, char *buff)
- Receive a reply to a control character Print a "printable" version of the character to the terminal Followed by a hex dump of the response.*

 - void [lspmac_GetmemReplyCB](#) (pmac_cmd_queue_t *cmd, int nreceived, char *buff)
- Service a reply to the getmem command.*

 - pmac_cmd_queue_t * [lspmac_SockGetmem](#) (int offset, int nbytes)
- Request a chunk of memory to be returned.*

 - pmac_cmd_queue_t * [lspmac_SockSendline](#) (char *event, char *fmt,...)
- Send a one line command.*

 - pmac_cmd_queue_t * [lspmac_SockSendline_nr](#) (char *event, char *fmt,...)
- Send a command and ignore the response.*

 - pmac_cmd_queue_t * [lspmac_SockSendControlCharPrint](#) (char *event, char c)
- Send a control character.*

 - void [lspmac_Getmem](#) ()
- Request a block of double buffer memory.*

 - void [lspmac_bo_read](#) (lspmac_motor_t *mp)
- Read the state of a binary i/o motor This is the read method for the binary i/o motor class.*

 - void [lspmac_dac_read](#) (lspmac_motor_t *mp)
- Read a DAC motor position.*

 - void [lspmac_shutter_read](#) (lspmac_motor_t *mp)
- Fast shutter read routine The shutter is mildly complicated in that we need to take into account the fact that the shutter can open and close again between status updates.*

 - void [lspmac_home1_queue](#) (lspmac_motor_t *mp)
- Home the motor.*

 - void [lspmac_home2_queue](#) (lspmac_motor_t *mp)
- Second stage of homing.*

 - double [lspmac_getPosition](#) (lspmac_motor_t *mp)
- get the motor position (with locking)*

 - void [lspmac_pmacmotor_read](#) (lspmac_motor_t *mp)
- Read the position and status of a normal PMAC motor.*

 - int [lspmac_getBIPosition](#) (lspmac_bi_t *bip)
- get binary input value*

 - void [lspmac_get_status_cb](#) (pmac_cmd_queue_t *cmd, int nreceived, char *buff)
- Service routing for status upate This updates positions and status information.*

 - void [lspmac_get_status](#) ()
- Request a status update from the PMAC.*

 - void [lspmac_more_ascii_cb](#) (pmac_cmd_queue_t *cmd, int nreceived, char *buff)
- we are expecting more characters from the DPRAM ASCII interface*

 - void [lspmac_get_ascii_cb](#) (pmac_cmd_queue_t *cmd, int nreceived, char *buff)
- service the ascii buffer request response*

 - void [lspmac_asciicmdCB](#) (pmac_cmd_queue_t *cmd, int nreceived, char *buf)
- PMAC has received our ascii command request Now see when it is ready for the next one.*

 - void [lspmac_SockSendDPline](#) (char *event, char *fmt,...)
- prepare (queue up) a line to send the dpram ascii command interface*

- void [lspmac_SockSendDPControlCharCB](#) ([pmac_cmd_queue_t](#) *cmd, int nreceived, char *buf)
- void [lspmac_SockSendDPControlChar](#) (char *event, char c)
use dpram ascii interface to send a control character
- void [lspmac_SockSendDPqueue](#) ()
- void [lspmac_abort](#) ()
abort motion and try to recover
- void [lspmac_GetAllIVarsCB](#) ([pmac_cmd_queue_t](#) *cmd, int nreceived, char *buff)
Receive the values of all the I variables Update our Postgresql database with the results.
- void [lspmac_GetAllIVars](#) ()
Request the values of all the I variables.
- void [lspmac_GetAllMVarsCB](#) ([pmac_cmd_queue_t](#) *cmd, int nreceived, char *buff)
Receive the values of all the M variables Update our database with the results.
- void [lspmac_GetAllMVars](#) ()
Request the values of all the M variables.
- void [lspmac_sendcmd_nocb](#) (char *fmt,...)
Send a command that does not need to deal with the reply.
- void [lspmac_sendcmd](#) (char *event, void(*responseCB)([pmac_cmd_queue_t](#) *, int, char *), char *fmt,...)
PMAC command with call back.
- void [lspmac_next_state](#) ()
State machine logic.
- void * [lspmac_worker](#) (void *dummy)
Our lspmac worker thread.
- int [lspmac_movedac_queue](#) ([lspmac_motor_t](#) *mp, double requested_position)
Move method for dac motor objects (ie, lights)
- int [lspmac_movezoom_queue](#) ([lspmac_motor_t](#) *mp, double requested_position)
Move method for the zoom motor.
- int [lspmac_move_preset_queue](#) ([lspmac_motor_t](#) *mp, char *preset_name)
Move a given motor to one of its preset positions.
- int [lspmac_test_preset](#) ([lspmac_motor_t](#) *mp, char *preset_name, double tolerance)
see if the motor is within tolerance of the preset 1 means yes, it is 0 mean no it isn't or that the preset was not found
- int [lspmac_moveabs_fshut_queue](#) ([lspmac_motor_t](#) *mp, double requested_position)
Move method for the fast shutter.
- int [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
- int [lspmac_moveabs_frontlight_oo_queue](#) ([lspmac_motor_t](#) *mp, double pos)
"move" frontlight on/off
- int [lspmac_moveabs_flight_factor_queue](#) ([lspmac_motor_t](#) *mp, double pos)
- int [lspmac_moveabs_blight_factor_queue](#) ([lspmac_motor_t](#) *mp, double pos)
- void [lspmac_video_rotate](#) (double secs)
Special motion program to collect centering video.
- double [lspmac_est_move_time](#) (double min_time, double delta, [lspmac_motor_t](#) *mp_1, char *preset_1, double end_point_1,...)
guess the minimum time it will take to move the motor
- int [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 Returns non-zero on abort, zero if OK.
- int [lspmac_move_or_jog_preset_queue](#) ([lspmac_motor_t](#) *mp, char *preset, int use_jog)
move using a preset value returns 0 on success, non-zero on error
- int [lspmac_moveabs_queue](#) ([lspmac_motor_t](#) *mp, double requested_position)
Use coordinate system motion program, if available, to move motor to requested position.

- `int lspmac_jogabs_queue (lspmac_motor_t *mp, double requested_position)`
Use jog to move motor to requested position.
- `int lspmac_moveabs_wait (lspmac_motor_t *mp, double timeout_secs)`
Wait for motor to finish moving.
- `void _lspmac_motor_init (lspmac_motor_t *d, char *name)`
Helper function for the init calls.
- `lspmac_motor_t * lspmac_motor_init (lspmac_motor_t *d, int wy, int wx, int *posp, int *stat1p, int *stat2p, char *wtitle, char *name, int(*moveAbs)(lspmac_motor_t *, double))`
Initialize a pmac stepper or servo motor.
- `lspmac_motor_t * lspmac_fshut_init (lspmac_motor_t *d)`
Initialize the fast shutter motor.
- `lspmac_motor_t * lspmac_bo_init (lspmac_motor_t *d, char *name, char *write_fmt, int *read_ptr, int read_mask)`
Initialize binary i/o motor.
- `lspmac_motor_t * lspmac_dac_init (lspmac_motor_t *d, int *posp, char *mvar, char *name, int(*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, int(*moveAbs)(lspmac_motor_t *, double))`
- `lspmac_bi_t * lspmac_bi_init (lspmac_bi_t *d, int *ptr, int mask, char *onEvent, char *offEvent)`
Initialize binary input.
- `void lspmac_init (int ivarsflag, int mvarsflag)`
Initialize this module.
- `void lspmac_cryoSwitchChanged_cb (char *event)`
- `void lspmac_scint_inPosition_cb (char *event)`
Maybe start drying off the scintillator.
- `void lspmac_backLight_up_cb (char *event)`
Turn on the backlight whenever it goes up.
- `void lspmac_backLight_down_cb (char *event)`
Turn off the backlight whenever it goes down.
- `void lspmac_light_zoom_cb (char *event)`
Set the backlight intensity whenever the zoom is changed (and the backlight is up)
- `void lspmac_scint_dried_cb (char *event)`
Turn off the dryer.
- `void lspmac_zoom_lut_setup ()`
Set up lookup table for zoom.
- `void lspmac_flight_lut_setup ()`
Set up lookup table for flight.
- `void lspmac_blight_lut_setup ()`
Set up lookup table for blight.
- `void lspmac_fscint_lut_setup ()`
Set up lookup table for fscint.
- `void lspmac_command_done_cb (char *event)`
- `void lspmac_run ()`
Start up the lspmac thread.

Variables

- static int `ls_pmac_state` = `LS_PMAC_STATE_DETACHED`
Current state of the PMAC communications state machine.
- int `lspmac_shutter_state`
State of the shutter, used to detect changes.
- int `lspmac_shutter_has_opened`
Indicates that the shutter had opened, perhaps briefly even if the state did not change.
- pthread_mutex_t `lspmac_shutter_mutex`
Coordinates threads reading shutter status.
- pthread_cond_t `lspmac_shutter_cond`
Allows waiting for the shutter status to change.
- pthread_mutex_t `lspmac_moving_mutex`
Coordinate moving motors between threads.
- pthread_cond_t `lspmac_moving_cond`
Wait for motor(s) to finish moving condition.
- int `lspmac_moving_flags`
Flag used to implement motor moving condition.
- static pthread_mutex_t `lspmac_ascii_mutex`
Keep too many processes from sending commands at once.
- static int `lspmac_ascii_busy` = 0
flag for condition to wait for
- static int `omega_zero_search` = 0
Indicate we'd really like to know when omega crosses zero.
- static double `omega_zero_velocity` = 0
rate (cnts/sec) that omega was traveling when it crossed zero
- struct timespec `omega_zero_time`
Time we believe that omega crossed zero.
- static struct timespec `lspmac_status_time`
Time the status was read.
- static struct timespec `lspmac_status_last_time`
Time the status was read.
- static pthread_t `pmac_thread`
our thread to manage access and communication to the pmac
- pthread_mutex_t `pmac_queue_mutex`
manage access to the pmac command queue
- pthread_cond_t `pmac_queue_cond`
wait for a command to be sent to PMAC before continuing
- static struct pollfd `pmacfd`
our poll structure
- static int `getivars` = 0
flag set at initialization to send i vars to db
- static int `getmvars` = 0
flag set at initialization to send m vars to db
- `lspmac_bi_t` `lspmac_bis` [32]
array of binary inputs
- int `lspmac_nbis` = 0
number of active binary inputs
- `lspmac_motor_t` `lspmac_motors` [48]
All our motors.
- int `lspmac_nmotors` = 0

- The number of motors we manage.*

 - `lspmac_motor_t * omega`

MD2 omega axis (the air bearing)
- `lspmac_motor_t * alignx`

Alignment stage X.
- `lspmac_motor_t * aligny`

Alignment stage Y.
- `lspmac_motor_t * alignz`

Alignment stage X.
- `lspmac_motor_t * anal`

Polaroid analyzer motor.
- `lspmac_motor_t * zoom`

Optical zoom.
- `lspmac_motor_t * apery`

Aperture Y.
- `lspmac_motor_t * aperz`

Aperture Z.
- `lspmac_motor_t * capy`

Capillary Y.
- `lspmac_motor_t * capz`

Capillary Z.
- `lspmac_motor_t * scint`

Scintillator Z.
- `lspmac_motor_t * cenx`

Centering Table X.
- `lspmac_motor_t * ceny`

Centering Table Y.
- `lspmac_motor_t * kappa`

Kappa.
- `lspmac_motor_t * phi`

Phi (not data collection axis)
- `lspmac_motor_t * fshut`

Fast shutter.
- `lspmac_motor_t * flight`

Front Light DAC.
- `lspmac_motor_t * blight`

Back Light DAC.
- `lspmac_motor_t * fscint`

Scintillator Piezo DAC.
- `lspmac_motor_t * smart_mag_oo`

Smart Magnet on/off.
- `lspmac_motor_t * blight_ud`

Back light Up/Down actuator.
- `lspmac_motor_t * cryo`

Move the cryostream towards or away from the crystal.
- `lspmac_motor_t * dryer`

blow air on the scintillator to dry it off
- `lspmac_motor_t * fluo`

Move the fluorescence detector in/out.
- `lspmac_motor_t * flight_oo`

Turn front light on/off.

- `lspmac_motor_t * blight_f`
Back light scale factor.
- `lspmac_motor_t * flight_f`
Front light scale factor.
- `lspmac_bi_t * lp_air`
Low pressure air OK.
- `lspmac_bi_t * hp_air`
High pressure air OK.
- `lspmac_bi_t * cryo_switch`
that little toggle switch for the cryo
- `lspmac_bi_t * blight_down`
Backlight is down.
- `lspmac_bi_t * blight_up`
Backlight is up.
- `lspmac_bi_t * cryo_back`
cryo is in the back position
- `lspmac_bi_t * fluor_back`
fluor is in the back position
- `lspmac_bi_t * sample_detected`
smart magnet detected sample
- `lspmac_bi_t * etel_ready`
ETEL is ready.
- `lspmac_bi_t * etel_on`
ETEL is on.
- `lspmac_bi_t * etel_init_ok`
ETEL initialized OK.
- `lspmac_bi_t * minikappa_ok`
Minikappa is OK (whatever that means)
- `lspmac_bi_t * smart_mag_on`
smart magnet is on
- `lspmac_bi_t * arm_parked`
(whose arm? parked where?)
- `lspmac_bi_t * shutter_open`
shutter is open (note in pmc says this is a slow input)
- `lspmac_bi_t * smart_mag_err`
smart magnet error (coil broken perhaps)
- `lspmac_bi_t * smart_mag_off`
smart magnet is off
- static unsigned char `dbmem` [64 * 1024]
double buffered memory
- static int `dbmemIn` = 0
next location
- static struct timeval
`pmac_time_sent` `now`
used to ensure we do not send commands to the pmac too often. Only needed for non-DB commands.
- static `pmac_cmd_t rr_cmd`
- static `pmac_cmd_t gb_cmd`
- static `pmac_cmd_t cr_cmd`
commands to send out "readready", "getbuffer", "controlresponse" (initialized in main)
- static `pmac_cmd_queue_t ethCmdQueue` [PMAC_CMD_QUEUE_LENGTH]
PMAC command queue.

- static unsigned int `ethCmdOn` = 0
points to next empty PMAC command queue position
- static unsigned int `ethCmdOff` = 0
points to current command (or none if == ethCmdOn)
- static unsigned int `ethCmdReply` = 0
Used like ethCmdOff only to deal with the pmac reply to a command.
- static char * `pmac_error_strs` []
Decode the errors perhaps returned by the PMAC.
- static `md2_status_t` `md2_status`
Buffer for MD2 Status.
- `pthread_mutex_t` `md2_status_mutex`
Synchronize reading/writing status buffer.
- static `lspmac_ascii_buffers_t` `lspmac_ascii_buffers`
- `pthread_mutex_t` `lspmac_ascii_buffers_mutex`
- static `lspmac_dpascii_queue_t` `lspmac_dpascii_queue` [`LSPMAC_DPASCII_QUEUE_LENGTH`]
- static `uint32_t` `lspmac_dpascii_on` = 0
- static `uint32_t` `lspmac_dpascii_off` = 0

7.6.1 Detailed Description

Routines concerned with communication with PMAC.

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

This is a state machine (surprise!) Lacking is support for writingbuffer, control writing and reading, as well as double buffered memory It looks like several different methods of managing PMAC communications are possible. Here is set up a queue of outgoing commands and deal completely with the result before sending the next. A full handshake of acknowledgements and "readready" is expected.

State	Description
-1	Reset the connection
0	Detached: need to connect to tcp port
1	Idle (waiting for a command to send to the pmac)
2	Send command
3	Waiting for command acknowledgement (no further response expected)
4	Waiting for control character acknowledgement (further response expected)
5	Waiting for command acknowledgement (further response expected)
6	Waiting for get memory response
7	Send controlresponse
8	Send readready
9	Waiting for acknowledgement of "readready"
10	Send readbuffer
11	Waiting for control response
12	Waiting for readbuffer response

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

7.6.2 Macro Definition Documentation

7.6.2.1 #define LS_PMAC_STATE_CR 7

Definition at line 45 of file `lspmac.c`.

7.6.2.2 `#define LS_PMAC_STATE_DETACHED 0`

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

7.6.2.3 `#define LS_PMAC_STATE_GB 10`

Definition at line 48 of file `lspmac.c`.

7.6.2.4 `#define LS_PMAC_STATE_GMR 6`

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

7.6.2.5 `#define LS_PMAC_STATE_IDLE 1`

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

7.6.2.6 `#define LS_PMAC_STATE_RESET -1`

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

7.6.2.7 `#define LS_PMAC_STATE_RR 8`

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

7.6.2.8 `#define LS_PMAC_STATE_SC 2`

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

7.6.2.9 `#define LS_PMAC_STATE_WACK 5`

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

7.6.2.10 `#define LS_PMAC_STATE_WACK_CC 4`

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

7.6.2.11 `#define LS_PMAC_STATE_WACK_NFR 3`

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

7.6.2.12 `#define LS_PMAC_STATE_WACK_RR 9`

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

7.6.2.13 `#define LS_PMAC_STATE_WCR 11`

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

7.6.2.14 #define LS_PMAC_STATE_WGB 12

Definition at line 50 of file lspmac.c.

7.6.2.15 #define LSPMAC_DPASCII_QUEUE_LENGTH 1024

Definition at line 357 of file lspmac.c.

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

Regex to pick out preset name and corresponding position.

Definition at line 140 of file lspmac.c.

7.6.2.17 #define PMAC_CMD_QUEUE_LENGTH 2048

Size of the PMAC command queue.

Definition at line 184 of file lspmac.c.

7.6.2.18 #define pmac_cmd_size 8

PMAC command size in bytes.

Definition at line 150 of file lspmac.c.

7.6.2.19 #define PMAC_MIN_CMD_TIME 10000.0

Minimum time between commands to the pmac.

Definition at line 180 of file lspmac.c.

7.6.2.20 #define PMACPORT 1025

The PMAC (only) listens on this port.

Definition at line 144 of file lspmac.c.

7.6.2.21 #define VR_CTRL_RESPONSE 0xc4

Definition at line 166 of file lspmac.c.

7.6.2.22 #define VR_DOWNLOAD 0x40

Definition at line 153 of file lspmac.c.

7.6.2.23 #define VR_FWDOWNLOAD 0xcb

Definition at line 170 of file lspmac.c.

7.6.2.24 #define VR_IPADDRESS 0xe0

Definition at line 171 of file lspmac.c.

7.6.2.25 #define VR_PMAC_FLUSH 0xb3

Definition at line 157 of file lspmac.c.

7.6.2.26 #define VR_PMAC_GETBUFFER 0xc5

Definition at line 167 of file lspmac.c.

7.6.2.27 #define VR_PMAC_GETLINE 0xb1

Definition at line 156 of file lspmac.c.

7.6.2.28 #define VR_PMAC_GETMEM 0xb4

Definition at line 158 of file lspmac.c.

7.6.2.29 #define VR_PMAC_GETRESPONSE 0xbf

Definition at line 164 of file lspmac.c.

7.6.2.30 #define VR_PMAC_PORT 0xbe

Definition at line 163 of file lspmac.c.

7.6.2.31 #define VR_PMAC_READREADY 0xc2

Definition at line 165 of file lspmac.c.

7.6.2.32 #define VR_PMAC_SENDCTRLCHAR 0xb6

Definition at line 160 of file lspmac.c.

7.6.2.33 #define VR_PMAC_SENDLINE 0xb0

Definition at line 155 of file lspmac.c.

7.6.2.34 #define VR_PMAC_SETBIT 0xba

Definition at line 161 of file lspmac.c.

7.6.2.35 #define VR_PMAC_SETBITS 0xbb

Definition at line 162 of file lspmac.c.

7.6.2.36 #define VR_PMAC_SETMEM 0xb5

Definition at line 159 of file lspmac.c.

7.6.2.37 #define VR_PMAC_WRITEBUFFER 0xc6

Definition at line 168 of file lspmac.c.

7.6.2.38 #define VR_PMAC_WRITEERROR 0xc7

Definition at line 169 of file lspmac.c.

7.6.2.39 #define VR_UPLOAD 0xc0

Definition at line 152 of file lspmac.c.

7.6.3 Typedef Documentation**7.6.3.1 typedef struct lspmac_ascii_buffers_struct lspmac_ascii_buffers_t****7.6.3.2 typedef struct lspmac_dpascii_queue_struct lspmac_dpascii_queue_t****7.6.3.3 typedef struct md2StatusStruct md2_status_t**

The block of memory retrieved in a status request.

7.6.4 Function Documentation**7.6.4.1 void _lspmac_motor_init(lspmac_motor_t * *d*, char * *name*)**

Helper function for the init calls.

Definition at line 2961 of file lspmac.c.

```

{
    lspmac_nmotors++;

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

    d->name = strdup(name);
    d->active = lsredis_get_obj( "%s.active",
        d->name);
    d->active_init = lsredis_get_obj( "
        %s.active_init", d->name);
    d->axis = lsredis_get_obj( "%s.axis",
        d->name);
    d->coord_num = lsredis_get_obj( "
        %s.coord_num", d->name);
    d->home = lsredis_get_obj( "%s.home",
        d->name);
    d->inactive_init = lsredis_get_obj( "
        %s.inactive_init", d->name);
    d->redis_fmt = lsredis_get_obj( "%s.format",
        d->name);
    d->max_accel = lsredis_get_obj( "
        %s.max_accel", d->name);
    d->max_speed = lsredis_get_obj( "
        %s.max_speed", d->name);
    d->max_pos = lsredis_get_obj( "
        %s.maxPosition", d->name);
    d->min_pos = lsredis_get_obj( "
        %s.minPosition", d->name);
    d->motor_num = lsredis_get_obj( "
        %s.motor_num", d->name);
    d->neg_limit_hit = lsredis_get_obj( "
        %s.negLimitSet", d->name);
    d->neutral_pos = lsredis_get_obj( "
        %s.neutralPosition", d->name);
    d->redis_position = lsredis_get_obj( "
        %s.position", d->name);

```

```

d->pos_limit_hit      = lsredis_get_obj( "
    %s.posLimitSet",    d->name);
d->precision          = lsredis_get_obj( "
    %s.precision",      d->name);
d->printf_fmt         = lsredis_get_obj( "
    %s.printf",         d->name);
d->status_str         = lsredis_get_obj( "
    %s.status_str",     d->name);
d->u2c                = lsredis_get_obj( "%s.u2c",
    d->name);
d->unit               = lsredis_get_obj( "%s.unit",
    d->name);
d->update_resolution  = lsredis_get_obj( "
    %s.update_resolution", d->name);
d->lut                 = NULL;
d->nlut               = 0;
d->homing              = 0;
d->dac_mvar            = NULL;
d->actual_pos_cnts_p   = NULL;
d->status1_p           = NULL;
d->status2_p           = NULL;
d->win                 = NULL;
d->read                = NULL;
d->reported_position  = INFINITY;
}

```

7.6.4.2 void cleanstr (char * s)

Replace \r with \n in null terminated string and print result to terminal.

Needed to turn PMAC messages into something printable.

Parameters

in	s	String to print to terminal.
----	---	------------------------------

Definition at line 537 of file lspmac.c.

```

{
    int i;

    pthread_mutex_lock( &ncurses_mutex);

    for( i=0; i<strlen( s); i++) {
        if( s[i] == '\r')
            wprintw( term_output, "\n");
        else
            wprintw( term_output, "%c", s[i]);
    }

    pthread_mutex_unlock( &ncurses_mutex);
}

```

7.6.4.3 void hex_dump (int n, unsigned char * s)

Prints a hex dump of the given data.

Used to debug packet data.

Parameters

in	n	Number of bytes passed in s
in	s	Data to dump

Definition at line 510 of file lspmac.c.

```

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

```

```

unsigned char outs[128], outsl[4];

for( i=0; n > 0; i++) {

    sprintf( (char *)outs, "%04d: ", 16*i);
    for( j=0; j<16 && n > 0; j++) {
        if( j==8)
            strcat( (char *)outs, " ");
        sprintf( (char *)outsl, " %02x", *(s + 16*i + j));
        strcat( (char *)outs, (char *)outsl);
        n--;
    }
    lslogging_log_message( "hex_dump: %s", outs);
}
}

```

7.6.4.4 void lsConnect (char * *ipaddr*)

Connect to the PMAC socket.

Establish or reestablish communications.

Parameters

in	<i>ipaddr</i>	String representation of the IP address (dot quad or FQN)
----	---------------	---

Definition at line 558 of file lspmac.c.

```

{
    int psock;                // our socket: value stored in pmacfd.fd
    int err;                  // error code from some system calls
    struct sockaddr_in *addrP; // our address structure to connect to
    struct addrinfo ai_hints;  // required for getaddrinfo
    struct addrinfo *ai_resultP; // linked list of address structures (we'll
                                // always pick the first)

    pmacfd.fd = -1;
    pmacfd.events = 0;

    // Initial buffer(s)
    memset( &ai_hints, 0, sizeof( ai_hints));

    ai_hints.ai_family = AF_INET;
    ai_hints.ai_socktype = SOCK_STREAM;

    //
    // get address
    //
    err = getaddrinfo( ipaddr, NULL, &ai_hints, &ai_resultP);
    if( err != 0) {

        lslogging_log_message( "Could not find address: %s",
                                gai_strerror( err));

        return;
    }

    addrP = (struct sockaddr_in *)ai_resultP->ai_addr;
    addrP->sin_port = htons( PMACPORT);

    psock = socket( PF_INET, SOCK_STREAM, 0);
    if( psock == -1) {
        lslogging_log_message( "Could not create socket");
        return;
    }

    err = connect( psock, (const struct sockaddr *)addrP, sizeof( *addrP));
    if( err != 0) {
        lslogging_log_message( "Could not connect socket: %s",
                                strerror( errno));
        return;
    }

    ls_pmac_state = LS_PMAC_STATE_IDLE;
    pmacfd.fd = psock;
    pmacfd.events = POLLIN;
}

```

```
}
```

7.6.4.5 void lspmac_abort ()

abort motion and try to recover

Definition at line 2012 of file lspmac.c.

```

{
//
// Stop everything! (consider ^O instead of ^A)
//
lspmac_SockSendDPControlChar( "Abort Request", 0
    x01);
}
```

7.6.4.6 void lspmac_asciicmdCB (pmac_cmd_queue_t * cmd, int nreceived, char * buf)

PMAC has received our ascii command request Now see when it is ready for the next one.

Definition at line 1943 of file lspmac.c.

```

{
lspmac_get_ascii( cmd->event);
}
```

7.6.4.7 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 3321 of file lspmac.c.

```

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

7.6.4.8 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 3314 of file lspmac.c.

```

{
blight->moveAbs( blight, round(lspmac_getPosition
    ( zoom)));
}
```

7.6.4.9 `lspmac_bi_t* lspmac_bi_init(lspmac_bi_t * d, int * ptr, int mask, char * onEvent, char * offEvent)`

Initialize binary input.

Definition at line 3116 of file `lspmac.c`.

```

    {
        lspmac_nbis++;
        pthread_mutex_init( &(d->mutex), NULL);
        d->ptr = ptr;
        d->mask = mask;
        d->changeEventOn = strdup( onEvent);
        d->changeEventOff = strdup( offEvent);
        d->first_time = 1;

        return d;
    }

```

7.6.4.10 `void lspmac_blight_lut_setup()`

Set up lookup table for blight.

Definition at line 3423 of file `lspmac.c`.

```

    {
        int i;
        lsredis_obj_t *p;

        pthread_mutex_lock( &blight->mutex);

        blight->nlut = 11;
        blight->lut = calloc( 2 * blight->nlut, sizeof( double));
        if( blight->lut == NULL) {
            lslogging_log_message( "lspmac_blight_lut_setup: out
                of memory");
            exit( -1);
        }

        blight->lut[0] = 0;
        blight->lut[1] = 0;

        for( i=1; i<blight->nlut; i++) {
            p = lsredis_get_obj( "cam.zoom.%d.LightIntensity", i);
            if( p==NULL || strlen( lsredis_getstr(p)) == 0) {
                free( blight->lut);
                blight->lut = NULL;
                blight->nlut = 0;
                pthread_mutex_unlock( &blight->mutex);
                lslogging_log_message( "lspmac_blight_lut_setup:
                    cannot find MotorPosition element for cam.blight level %d", i);
                return;
            }
            blight->lut[2*i] = i;
            blight->lut[2*i+1] = 20000.0 * lsredis_getd( p) / 100.
                0;
        }
        for( i=0; i<blight->nlut; i++) {
            lslogging_log_message( "lspmac_blight_lut_setup: i:
                %d x: %f y: %f y(lut): %f x(rlut): %f",
                    i, blight->lut[2*i], blight->lut[2
                        *i+1],
                    lspmac_lut( blight->nlut, blight
                        ->lut, blight->lut[2*i]),
                    lspmac_rlut( blight->nlut,
                        blight->lut, blight->lut[2*i+1])
                    );
        }
        pthread_mutex_unlock( &blight->mutex);
    }

```

7.6.4.11 `lspmac_motor_t* lspmac_bo_init(lspmac_motor_t * d, char * name, char * write_fmt, int * read_ptr, int read_mask)`

Initialize binary i/o motor.

Parameters

in	<i>d</i>	Our uninitialized motor object
in	<i>name</i>	Name of motor to coordinate with DB
in	<i>write_fmt</i>	Format string used to generate PMAC command to move motor
in	<i>read_ptr</i>	Pointer to byte in md2_status to find position
in	<i>read_mask</i>	Bitmask to find position in *read_ptr

Definition at line 3050 of file lspmac.c.

```

{
    _lspmac_motor_init( d, name);

    d->moveAbs      = lspmac_moveabs_bo_queue;
    d->read          = lspmac_bo_read;
    d->write_fmt     = strdup( write_fmt);
    d->read_ptr      = read_ptr;
    d->read_mask     = read_mask;

    return d;
}

```

7.6.4.12 void lspmac_bo_read (lspmac_motor_t * mp)

Read the state of a binary i/o motor This is the read method for the binary i/o motor class.

Parameters

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

Definition at line 1121 of file lspmac.c.

```

{
    int pos, changed;

    pthread_mutex_lock( &(mp->mutex));

    pos = (*(mp->read_ptr) & mp->read_mask) == 0 ? 0 : 1;

    changed = pos != mp->position;
    mp->position = pos;

    pthread_mutex_unlock( &(mp->mutex));

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

```

7.6.4.13 void lspmac_command_done.cb (char * event)

Definition at line 3483 of file lspmac.c.

```

{
    int i;
    lspmac_motor_t *mp;

    // O(n). Bad.
    //
    for( i=0; i<lspmac_nmotors; i++) {
        if( strcmp( lspmac_motors[i].name, event, strlen(
            lspmac_motors[i].name)) == 0)
            break;
    }

    if( i >= lspmac_nmotors)
        return;

    mp = &(lspmac_motors[i]);
    pthread_mutex_lock( &(mp->mutex));
}

```



```

mp->command_sent = 1;

pthread_cond_signal( &(mp->cond));
pthread_mutex_unlock( &(mp->mutex));

return;
}

```

7.6.4.14 void lspmac_cryoSwitchChanged_cb (char * event)

Definition at line 3276 of file lspmac.c.

```

{

int pos;

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

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

```

7.6.4.15 lspmac_motor_t* lspmac_dac_init (lspmac_motor_t * d, int * posp, char * mvar, char * name, int (*)(lspmac_motor_t *, double) moveAbs)

Initialize DAC motor Note that some motors require further initialization from a database query.

For this reason this initialization code must be run before the database queue is allowed to be processed.

Parameters

out	<i>d</i>	Returns the (almost) initialized motor object [in,out] uninitialized motor
in	<i>posp</i>	Location of current position
in	<i>mvar</i>	M variable, ie, "M1200"
in	<i>name</i>	name to coordinate with DB
in	<i>moveAbs</i>	Method to use to move this motor

Definition at line 3076 of file lspmac.c.

```

{

_lspmac_motor_init( d, name);
d->moveAbs          = moveAbs;
d->read              = lspmac_dac_read;
d->actual_pos_cnts_p = posp;
d->dac_mvar          = strdup(mvar);

return d;
}

```

7.6.4.16 void lspmac_dac_read (lspmac_motor_t * mp)

Read a DAC motor position.

Parameters

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

Definition at line 1141 of file lspmac.c.

```

{

double u2c;

```

```

pthread_mutex_lock( &(mp->mutex));
mp->actual_pos_cnts = *mp->actual_pos_cnts_p;
u2c = lsredis_getd( mp->u2c);

if( mp->nlut >0 && mp->lut != NULL) {
    if( u2c == 0.0)
        u2c = 1.0;
    mp->position = lspmac_rlut( mp->nlut, mp->lut, mp
        ->actual_pos_cnts/u2c);
} else {
    if( u2c != 0.0) {
        mp->position = mp->actual_pos_cnts / u2c;
    } else {
        mp->position = mp->actual_pos_cnts;
    }
}

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

```

7.6.4.17 void lspmac_Error (char * buff)

The service routing detected an error condition.

Scan the response buffer for an error code and print it out.

Parameters

in	buff	Buffer returned by PMAC perhaps containing a NULL terminated message.
----	------	---

Definition at line 767 of file lspmac.c.

```

{
    int err;
    //
    // assume buff points to a 1400 byte array of stuff read from the pmac
    //

    if( buff[0] == 7 && buff[1] == 'E' && buff[2] == 'R' && buff[3] == 'R') {
        buff[7] = 0; // For null termination
        err = atoi( &(buff[4]));
        if( err > 0 && err < 20) {
            lslogging_log_message( pmac_error_strs
                [err]);

            pthread_mutex_lock( &ncurses_mutex);
            wprintw( term_output, "\n%s\n", pmac_error_strs
                [err]);
            wnoutrefresh( term_output);
            wnoutrefresh( term_input);
            doupdate();
            pthread_mutex_unlock( &ncurses_mutex);
        }
    }
    lspmac_Reset();
}

```

7.6.4.18 double lspmac_est_move_time (double min_time, double delta, lspmac_motor_t * mp_1, char * preset_1, double end_point_1, ...)

guess the minimum time it will take to move the motor

Parameters

min_time	Return at least this value. Can be zero if you like.
delta	Add this fudge factor to the estimate so we always over estimate the time (You do not want to under estimate or even be exactly right)
mp_1	Pointer to first motor
preset_1	Name of preset we'd like to move to or NULL if end_point_1 should be used instead

<code>end_point_1</code>	End point for the first motor. Ignored if preset_1 is non null and identifies a valid preset for this motor
<code>...</code>	Perhaps more pairs of motors and end points MUST END ARG LIST WITH NULL

Definition at line 2606 of file lspmac.c.

```

{
    va_list arg_ptr;
    lspmac_motor_t *mp;
    double ep, maybe_ep;
    char *ps;
    double rtn;
    double t;
    int err;

    rtn = min_time;
    mp = mp_1;
    ps = preset_1;
    ep = end_point_1;

    va_start( arg_ptr, end_point_1);
    while( 1 ) {

        t = min_time;
        if( mp != NULL && mp->max_speed != NULL && mp->max_accel
            != NULL && mp->u2c &&
            lsredis_getd( mp->max_speed) > 0 && lsredis_getd
            (mp->max_accel) > 0 && lsredis_getd( mp->u2c) > 0 ) {

            if( ps != NULL && *ps != 0 ) {
                err = lsredis_find_preset( mp->name, ps, &
                    maybe_ep);
                if( err != 0)
                    ep = maybe_ep;
            }

            // time if we went the entire way at constant velocity
            //
            t = fabs(ep - lspmac_getPosition(mp)) / lsredis_getd
            ( mp->u2c) / lsredis_getd( mp->max_speed);

            // Correction for acceleration time
            //
            t += lsredis_getd( mp->max_speed) / lsredis_getd
            ( mp->max_accel);

            // Add something for good measure: we always want to overestimate the
            timeout
            t += delta;

        }

        rtn = rtn < t ? t : rtn;

        mp = va_arg( arg_ptr, lspmac_motor_t *);
        if( mp == NULL)
            break;

        ps = va_arg( arg_ptr, char *);
        ep = va_arg( arg_ptr, double);
    }
    va_end( arg_ptr);

    return rtn;
}

```

7.6.4.19 void lspmac_flight_lut_setup ()

Set up lookup table for flight.

Definition at line 3390 of file lspmac.c.

```

{
    int i;
    lsredis_obj_t *p;

```

```

pthread_mutex_lock( &flight->mutex);

flight->nlut = 11;
flight->lut = calloc( 2 * flight->nlut, sizeof( double));
if( flight->lut == NULL) {
    lslogging_log_message( "lspmac_flight_lut_setup: out
        of memory");
    exit( -1);
}

flight->lut[0] = 0;
flight->lut[1] = 0;
for( i=1; i < flight->nlut; i++) {
    p = lsredis_get_obj( "cam.zoom.%d.FrontLightIntensity", i);
    if( p==NULL || strlen( lsredis_getstr(p)) == 0) {
        free( flight->lut);
        flight->lut = NULL;
        flight->nlut = 0;
        pthread_mutex_unlock( &flight->mutex);
        lslogging_log_message( "lspmac_flight_lut_setup:
            cannot find MotorPosition element for cam.flight level %d", i);
        return;
    }
    flight->lut[2*i] = i;
    flight->lut[2*i+1] = 32767.0 * lsredis_getd( p) / 100.
        0;
}
pthread_mutex_unlock( &flight->mutex);
}

```

7.6.4.20 void lspmac_fscint_lut_setup ()

Set up lookup table for fscint.

Definition at line 3464 of file lspmac.c.

```

{
    int i;

    pthread_mutex_lock( &fscint->mutex);

    fscint->nlut = 101;
    fscint->lut = calloc( 2 * fscint->nlut, sizeof( double));
    if( fscint->lut == NULL) {
        lslogging_log_message( "lspmac_fscint_lut_setup: out
            of memory");
        exit( -1);
    }

    for( i=0; i<fscint->nlut; i++) {
        fscint->lut[2*i] = i;
        fscint->lut[2*i+1] = 320.0 * i;
    }
    pthread_mutex_unlock( &fscint->mutex);
}

```

7.6.4.21 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 3034 of file lspmac.c.

```

{
    _lspmac_motor_init( d, "fastShutter");

    d->moveAbs = lspmac_moveabs_fshut_queue;
    ;
    d->read = lspmac_shutter_read;
}

```

```

    return d;
}

```

7.6.4.22 void lspmac_get_ascii (char * event)

Forward declaretion.

Request the ascii buffers from the PMAC.

Definition at line 1935 of file lspmac.c.

```

    {
        lspmac_send_command( VR_UPLOAD, VR_PMAC_GETMEM
            , 0x0e9c, 0, sizeof(lspmac_ascii_buffers_t), NULL,
            lspmac_get_ascii_cb, 0, event);
    }

```

7.6.4.23 void lspmac_get_ascii_cb (pmac_cmd_queue_t * cmd, int nreceived, char * buff)

service the ascii buffer request response

Definition at line 1827 of file lspmac.c.

```

uint32_t clrdata;
int need_more;

need_more = 0;
pthread_mutex_lock( &lspmac_ascii_mutex);
memcpy( &lspmac_ascii_buffers, buff, sizeof(
    lspmac_ascii_buffers));

//
// The response is not ready yet
// This will be an infinite loop if we queue a command that does not
// produce a response.
//
// Quoted comments below from Delta Tau "Turbo PMAC User Manual 9/12/2008,
// page 422"
//
// "1. Wait for the Host-Input Control Word at 0x0F40 (Y:$063D0) to become
// greater than 0, indicating
// that a response line is ready."
//
if( lspmac_ascii_buffers.response_buf == 0) {
    need_more = 1;
} else {
    if( (lspmac_ascii_buffers.response_buf & 0
        x8000) != 0) {
        char bcd1, bcd2, bcd3;
        int errcode;
        // Error response
        //
        // "2. Interpret the value in this register to determine what
        // type of response is present. If Bit 15 is 1, Turbo PMAC is
        // reporting an error in the command, and there is no response
        // other than this word. In this case, Bits 0 - 11 encode the
        // error number for the command as 3 BCD digits."
        //
        need_more = 0;
        bcd1 = lspmac_ascii_buffers.response_buf
            & 0x000f;
        bcd2 = (lspmac_ascii_buffers.response_buf
            & 0x00f0) >> 4;
        bcd3 = (lspmac_ascii_buffers.response_buf
            & 0x0f00) >> 8;
        errcode = (bcd3 * 10 + bcd2) * 10 + bcd1;

        if( errcode >= sizeof( pmac_error_strs)/sizeof(
            *pmac_error_strs))
            errcode = 0;
        lslogging_log_message( "lspmac_get_ascii_cb: Error
            returned for %s: %s", lspmac_ascii_buffers.command_str
            , pmac_error_strs[errcode]);
        //

```

```

        // Command not allowed during program execution.
        //
        // Requeue it;
        if( errcode == 1) {
            lspmac_dpascii_off--;
        }
    } else {
        //
        // "3. Read the response string starting at 0x0F44
        // (Y:$0603D1). Two 8-bit characters are packed into each 16-bit
        // word; the first character is placed into the low
        // byte. Subsequent characters are placed into consecutive
        // higher addresses, two per 16-bit word. (In byte addressing,
        // each character is read from an address one higher than the
        // preceding character.) Up to 255 characters can be sent in a
        // single response line. The string is terminated with the NULL
        // character (byte value 0), convenient for C-style string
        // handling. For Pascal-style string handling, the register at
        // 0x0F42 (X:$0603D0) contains the number of characters in the
        // string (plus one)."
```

```

        //
        if( lspmac_ascii_buffers.response_n > 1)
            lslogging_log_message( "lspmac_get_ascii_cb: '%s'
            '%s'", lspmac_ascii_buffers.command_str,
            lspmac_ascii_buffers.response_str);
        else
            lslogging_log_message( "lspmac_get_ascii_cb: '%s'
            responded", lspmac_ascii_buffers.command_str);

        //
        // 5. "If Bits 0 - 7 of the Host-Input Control Word had
        // contained the value $0D (13 decimal, "CR"), this was not the
        // last line in the response, and steps 1 - 4 should be
        // repeated. If they had contained the value $06 (6 decimal,
        // "ACK"), this was the last line in the response."
        //
        if( (lspmac_ascii_buffers.response_buf &
        0x00ff) == 0x0d) {
            need_more = 1;
        } else {
            need_more = 0;

            if( cmd->event != NULL && *(cmd->event) != 0)
                lsevents_send_event( "%s command accepted", cmd->
                event);
        }
    }
}

pthread_mutex_unlock( &lspmac_ascii_mutex);

//
// Reset the buffer flags and, perhaps, requeue a request
//
// "4. Clear the Host-Input Control Word at 0x0F40 (Y:$063D0)
// to 0. Turbo PMAC will not send another response line until it sees
// this register set to 0."
//
clrdata = 0;          // set the control word to zero

if( need_more) {
    lspmac_send_command( VR_UPLOAD, VR_PMAC_SETMEM
        , 0x0f40, 0, 4, (char *)&clrdata, lspmac_more_ascii_cb, 1,
        NULL);
} else {
    lspmac_send_command( VR_UPLOAD, VR_PMAC_SETMEM
        , 0x0f40, 0, 4, (char *)&clrdata, NULL, 1, NULL);
    lspmac_ascii_busy = 0;
}
}

```

7.6.4.24 void lspmac_get_status ()

Request a status update from the PMAC.

Definition at line 1815 of file lspmac.c.

```

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

```

7.6.4.25 void lspmac_get_status_cb (pmac_cmd_queue_t * cmd, int nreceived, char * buff)

Service routing for status upate This updates positions and status information.

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 1591 of file lspmac.c.

```

    {
#ifdef SHOW_RATE
static struct timespec ts1;
static struct timespec ts2;
static int cnt = 0;
#endif

int i;
lspmac_bi_t *bp;

clock_gettime( CLOCK_REALTIME, &lspmac_status_time);

#ifdef SHOW_RATE
if( cnt == 0) {
    clock_gettime( CLOCK_REALTIME, &ts1);
}
#endif

pthread_mutex_lock( &md2_status_mutex);
memcpy( &md2_status, buff, sizeof(md2_status));
//
// Note that we are the only thread that writes to md2_status
// so we no longer need the lock to read. Other threads must
// lock the mutex to read md2_status.
//
pthread_mutex_unlock( &md2_status_mutex);

//
// track the coordinate system moving flags
//
pthread_mutex_lock( &lspmac_moving_mutex);
if( md2_status.moving_flags != lspmac_moving_flags
) {
    int mask;

    lslogging_log_message( "lspmac_get_status_cb: new
        moving flag: %0x", md2_status.moving_flags);
    mask = 1;
    for( i=1; i<=16; i++, mask <= 1) {
        if( ((lspmac_moving_flags & mask) != 0) && ((
            md2_status.moving_flags & mask) == 0)) {
            // Falling edge: send event
            lsevents_send_event( "Coordsys %d Stopped", i);
        }
    }
    lspmac_moving_flags = md2_status.moving_flags
;
    pthread_cond_signal( &lspmac_moving_cond);
}
pthread_mutex_unlock( &lspmac_moving_mutex);

//
// Read the motor positions
//
for( i=0; i<lspmac_nmotors; i++) {
    lspmac_motors[i].read(&(lspmac_motors[i]));
}

//
// Read the binary inputs and perhaps send an event
//
for( i=0; i<lspmac_nbis; i++) {
    bp = &(lspmac_bis[i]);

    pthread_mutex_lock( &(bp->mutex));

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

```

```

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

pthread_mutex_lock( &ncurses_mutex);

// acc11c_1  INPUTS
// mask  bit
// 0x01  0   M1000   Air pressure OK
// 0x02  1   M1001   Air bearing OK
// 0x04  2   M1002   Cryo switch
// 0x08  3   M1003   Backlight Down
// 0x10  4   M1004   Backlight Up
// 0x20  5
// 0x40  6   M1006   Cryo is back

//
// acc11c_2  INPUTS
// mask  bit
// 0x01  0   M1008   Fluor Dector back
// 0x02  1   M1009   Sample Detected
// 0x04  2   M1020   {SC load request}
// 0x08  3   M1021   {SC move cryo back request}
// 0x10  4   M1022   {SC sample magnet control}
// 0x20  5   M1013   Etel Ready
// 0x40  6   M1014   Etel On
// 0x80  7   M1015   Etel Init OK

if( md2_status.acc11c_2 & 0x01)
    mvwprintw( term_status2, 3, 10, "%s", -8, "Fluor Out");
else
    mvwprintw( term_status2, 3, 10, "%s", -8, "Fluor In");

if( md2_status.acc11c_5 & 0x08)
    mvwprintw( term_status2, 4, 1, "%s", -(LS_DISPLAY_WINDOW_WIDTH
        -2), "Dryer On");
else
    mvwprintw( term_status2, 4, 1, "%s", -(LS_DISPLAY_WINDOW_WIDTH
        -2), "Dryer Off");

if( md2_status.acc11c_2 & 0x02)
    mvwprintw( term_status2, 2, 1, "%s", -(LS_DISPLAY_WINDOW_WIDTH
        -2), "Cap Dected");
else
    mvwprintw( term_status2, 2, 1, "%s", -(LS_DISPLAY_WINDOW_WIDTH
        -2), "Cap Not Dected");
wnoutrefresh( term_status2);

// acc11c_3  INPUTS
// mask  bit
// 0x01  0   M1025   Minikappa OK
// 0x02  1   M1023   {SC unload request}
// 0x04  2   M1024   Smartmagnet is on (note in pmc saying this is not used
//                  in VB interface)
// 0x08  3   M1027   Arm Parked
// 0x10  4   M1031   Smartmagnet error (coil is broken)
// 0x20  5
// 0x40  6
// 0x80  7
// 0x100 8   M1048   Shutter is open (note in pmc says: slow input !!!)

// acc11c_4  INPUTS
// mask  bit
// 0x01  0   M1031   {laser mirror is back}

```



```

// 0x02 1 M1032 {laser PSS OK}
// 0x04 2 M1033 {laser shutter open}

// acc11c_5 OUTPUTS
// mask bit
// 0x01 0 M1100 Mag Off
// 0x02 1 M1191 Condenser Out
// 0x04 2 M1102 Cryo Back
// 0x08 3 M1103 Dryer On
// 0x10 4 M1104 FluoDet Out
// 0x20 5 M1105 {smartmagnet on/off: note in pmc says this is not used}
// 0x40 6 M1106 1=SmartMag, 0=Permanent Mag
//

if( md2_status.acc11c_5 & 0x04)
    mvwprintw( term_status2, 3, 1, "%s", -8, "Cryo Out");
else
    mvwprintw( term_status2, 3, 1, "%s", -8, "Cryo In ");

// acc11c_6 OUTPUTS
// mask bit
// 0x0001 0 M1040 {SC Sample transfer is on}
// 0x0002 1
// 0x0004 2
// 0x0008 3
// 0x0010 4
// 0x0020 5
// 0x0040 6
// 0x0080 7 M1115 Etel Enable
// 0x0100 8 M1124 Fast Shutter Enable
// 0x0200 9 M1125 Fast Shutter Manual Enable
// 0x0400 10 M1126 Fast Shutter On
// 0x0800 11
// 0x1000 12 M1128 ADC1 gain bit 0
// 0x2000 13 M1129 ADC1 gain bit 1
// 0x4000 14 M1130 ADC2 gain bit 0
// 0x8000 15 M1131 ADC2 gain bit 1
//

if( md2_status.acc11c_5 & 0x02)
    mvwprintw( term_status, 3, 1, "%s", -(LS_DISPLAY_WINDOW_WIDTH
-2), "Backlight Up");
else
    mvwprintw( term_status, 3, 1, "%s", -(LS_DISPLAY_WINDOW_WIDTH
-2), "Backlight Down");

mvwprintw( term_status, 4, 1, "Front: %u",
    LS_DISPLAY_WINDOW_WIDTH-2-8, (int)flight->position);
mvwprintw( term_status, 5, 1, "Back: %u", LS_DISPLAY_WINDOW_WIDTH
-2-7, (int)blight->position);
mvwprintw( term_status, 6, 1, "Piezo: %u",
    LS_DISPLAY_WINDOW_WIDTH-2-8, (int)fscint->position);
wnoutrefresh( term_status);

wnoutrefresh( term_input);
doupdate();
pthread_mutex_unlock( &ncurses_mutex);

#ifdef SHOW_RATE
if( ++cnt % 1000 == 0) {
    long diff_sec;
    long diff_nsec;

    clock_gettime( CLOCK_REALTIME, &ts2);

    diff_sec = ts2.tv_sec - ts1.tv_sec;
    diff_nsec = ts2.tv_nsec - ts1.tv_nsec;

    if( diff_nsec < 0) {
        diff_nsec += 1000000000;
        diff_sec--;
    }

    lslogging_log_message( "Refresh Rate: %0.1f Hz", (
        double)cnt / (diff_sec + diff_nsec/1000000000.));

    cnt = 0;
}
#endif
}

```

7.6.4.26 void lspmac.GetAllIVars ()

Request the values of all the I variables.

Definition at line 2042 of file lspmac.c.

```

    {
static char *cmds = "I0..8191";
lspmac_send_command( VR_DOWNLOAD,
VR_PMAC_SENDLINE, 0, 0, strlen( cmds), cmds,
lspmac_GetAllIVarsCB, 0, NULL);
}

```

7.6.4.27 void lspmac.GetAllIVarsCB (pmac_cmd_queue_t * cmd, int nreceived, char * buff)

Receive the values of all the I variables Update our Postgresql database with the results.

Parameters

in	<i>cmd</i>	The command that gave this response
in	<i>nreceived</i>	Number of bytes received
in	<i>buff</i>	The byte buffer

Definition at line 2025 of file lspmac.c.

```

    {
static char qs[LS_PG_QUERY_STRING_LENGTH];
char *sp;
int i;
for( i=0, sp=strtok( buff, "\r"); sp != NULL; sp=strtok( NULL, "\r"), i++) {
    snprintf( qs, sizeof( qs)-1, "SELECT pmac.md2_ivar_set( %d, '%s')", i, sp);
    qs[sizeof( qs)-1]=0;
    lspg_query_push( NULL, qs);
}
}

```

7.6.4.28 void lspmac.GetAllMVars ()

Request the values of all the M variables.

Definition at line 2067 of file lspmac.c.

```

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

```

7.6.4.29 void lspmac.GetAllMVarsCB (pmac_cmd_queue_t * cmd, int nreceived, char * buff)

Receive the values of all the M variables Update our database with the results.

Parameters

in	<i>cmd</i>	The command that started this
in	<i>nreceived</i>	Number of bytes received
in	<i>buff</i>	Our byte buffer

Definition at line 2050 of file lspmac.c.

```

{

```

```

static char qs[LS_PG_QUERY_STRING_LENGTH];
char *sp;
int i;
for( i=0, sp=strtok(buff, "\\r"); sp != NULL; sp=strtok( NULL, "\\r"), i++) {
    snprintf( qs, sizeof( qs)-1, "SELECT pmac.md2_mvar_set( %d, '%s'", i, sp);
    qs[sizeof( qs)-1]=0;
    lspg_query_push( NULL, qs);
}
}

```

7.6.4.30 int lspmac_getBIPosition (lspmac_bi_t * bip)

get binary input value

Definition at line 1579 of file lspmac.c.

```

{
    int rtn;
    pthread_mutex_lock( &bip->mutex);
    rtn = bip->position;
    pthread_mutex_unlock( &bip->mutex);
    return rtn;
}

```

7.6.4.31 void lspmac_Getmem ()

Request a block of double buffer memory.

Definition at line 1112 of file lspmac.c.

```

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

```

7.6.4.32 void lspmac_GetmemReplyCB (pmac_cmd_queue_t * cmd, int nreceived, char * buff)

Service a reply to the getmem command.

Parameters

<i>cmd</i>	Queue item this is a reply to
<i>nreceived</i>	Number of bytes received
<i>buff</i>	Buffer of bytes recieved

Definition at line 1038 of file lspmac.c.

```

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

```

7.6.4.33 double lspmac_getPosition (lspmac_motor_t * mp)

get the motor position (with locking)

Parameters

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

Definition at line 1339 of file lspmacc.c.

```

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

```

7.6.4.34 void lspmacc_GetShortReplyCB (pmac_cmd_queue_t * cmd, int nreceived, char * buff)

Receive a reply that does not require multiple buffers.

Parameters

in	<i>cmd</i>	Queue item this is a reply to
in	<i>nreceived</i>	Number of bytes received
in	<i>buff</i>	The buffer of bytes

Definition at line 981 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));
}

```

7.6.4.35 void lspmacc_home1_queue (lspmacc_motor_t * mp)

Home the motor.

Parameters

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

Definition at line 1205 of file lspmacc.c.

```

{
    int i;
    int motor_num;
    int coord_num;
    char **home;
}

```

```

pthread_mutex_lock( &(mp->mutex));

motor_num = lsredis_getl( mp->motor_num);
coord_num = lsredis_getl( mp->coord_num);
home      = lsredis_get_string_array( mp->home);

// Each of the motors should have this defined
// but let's not seg fault if home is missing
//
if( home == NULL || *home == NULL) {
    //
    // Note we are already initialized
    // so if we are here there is something wrong.
    //
    lslogging_log_message( "lspmac_home1_queue: null or
        empty home strings for motor %s", mp->name);
    pthread_mutex_unlock( &(mp->mutex));
    return;
}

// We've already been called. Don't home again until
// we're finish with the last time.
//
if( mp->homing) {
    pthread_mutex_unlock( &(mp->mutex));
    return;
}

//
// Don't go on if any other motors in this coordinate system are homing.
// It's possible to write the homing program to home all the motors in the
// coordinate
// system. TODO (hint hint)
//
if( coord_num > 0) {
    for( i=0; i<lspmac_nmotors; i++) {
        if( &(lspmac_motors[i]) == mp)
            continue;
        if( lsredis_getl(lspmac_motors[i].coord_num) ==
            coord_num) {
            int nogo;
            nogo = 0;
            pthread_mutex_lock( &(lspmac_motors[i].mutex));
            //
            // Don't go on if
            //
            // we are homing      or      ( not in position
            while      in open loop)
            //
            if( lspmac_motors[i].homing || ((lspmac_motors
            [i].status2 & 0x01)==0) && ((lspmac_motors[i].status1 & 0x040000)
            != 0)))
                nogo = 1;
            pthread_mutex_unlock( &(lspmac_motors[i].mutex));
            if( nogo) {
                pthread_mutex_unlock( &(mp->mutex));
                return;
            }
        }
    }
}

mp->homing = 1;
mp->not_done = 1;    // set up waiting for cond
mp->motion_seen = 0;
// This opens the control loop.
// The status routine should notice this and the fact that
// the homing flag is set and call on the home2 routine
//
// Only send the open loop command if we are not in
// open loop mode already. This test might prevent a race condition
// where we've already moved the home2 routine (and queue the homing program
// motion)
// before the open loop command is dequeued and acted on.
//
if( ~(mp->status1) & 0x040000) {
    lspmac_SockSendDPLine( mp->name, "##d$*",
        motor_num);
}

pthread_mutex_unlock( &(mp->mutex));

lsevents_send_event( "%s Homing", mp->name);
}

```

7.6.4.36 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 1293 of file lspmac.c.

```

{

char **spp;
char **home;

//
// At this point we are in open loop.
// Run the motor specific commands
//

pthread_mutex_lock( &(mp->mutex));

home = lsredis_get_string_array( mp->home);

//
// We don't have any motors that have a null home text array so
// there is currently no need to worry about this case other than
// not to seg fault
//
// Also, Only go on if the first homing phase has been started
//
if( home == NULL || mp->homing != 1) {
    pthread_mutex_unlock( &(mp->mutex));
    return;
}

for( spp = home; *spp != NULL; spp++) {

    pthread_mutex_lock( &ncurses_mutex);
    wprintw( term_output, "home2 is queuing '%s'\n", *spp);
    wnoutrefresh( term_output);
    doupdate();
    pthread_mutex_unlock( &ncurses_mutex);

    lspmac_SockSendDPLine( mp->name, *spp);
}

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

```

7.6.4.37 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 3132 of file lspmac.c.

```

{
md2_status_t *p;
pthread_mutexattr_t mutex_initializer;

// Set our global harvest flags
getivars = ivarsflag;
getmvars = mvarsflag;

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

```

```

// All important status mutex
pthread_mutex_init( &md2_status_mutex, &mutex_initializer);

//
// Get the MD2 initialization strings
//
//  lspmac_md2_init = lsredis_get_obj( "md2_pmac.init"); // hard coded now.

//
// Initialize the motor objects
//

p = &md2_status;

omega = lspmac_motor_init( &(lspmac_motors
[ 0]), 0, 0, &p->omega_act_pos, &p->omega_status_1
, &p->omega_status_2, "Omega #1 &1 X", "omega",
lspmac_moveabs_queue);
alignx = lspmac_motor_init( &(lspmac_motors
[ 1]), 0, 1, &p->alignx_act_pos, &p->alignx_status_1
, &p->alignx_status_2, "Align X #2 &3 X", "align.x",
lspmac_moveabs_queue);
aligny = lspmac_motor_init( &(lspmac_motors
[ 2]), 0, 2, &p->aligny_act_pos, &p->aligny_status_1
, &p->aligny_status_2, "Align Y #3 &3 Y", "align.y",
lspmac_moveabs_queue);
alignz = lspmac_motor_init( &(lspmac_motors
[ 3]), 0, 3, &p->alignz_act_pos, &p->alignz_status_1
, &p->alignz_status_2, "Align Z #4 &3 Z", "align.z",
lspmac_moveabs_queue);
anal = lspmac_motor_init( &(lspmac_motors
[ 4]), 0, 4, &p->analyzer_act_pos, &p->analyzer_status_1
, &p->analyzer_status_2, "Anal #5", "lightPolar",
lspmac_moveabs_queue);
zoom = lspmac_motor_init( &(lspmac_motors
[ 5]), 1, 0, &p->zoom_act_pos, &p->zoom_status_1
, &p->zoom_status_2, "Zoom #6 &4 Z", "cam.zoom",
lspmac_movezoom_queue);
apery = lspmac_motor_init( &(lspmac_motors
[ 6]), 1, 1, &p->aperturey_act_pos, &p->aperturey_status_1
, &p->aperturey_status_2, "Aper Y #7 &5 Y", "appy",
lspmac_moveabs_queue);
aperez = lspmac_motor_init( &(lspmac_motors
[ 7]), 1, 2, &p->aperturez_act_pos, &p->aperturez_status_1
, &p->aperturez_status_2, "Aper Z #8 &5 Z", "appz",
lspmac_moveabs_queue);
capy = lspmac_motor_init( &(lspmac_motors
[ 8]), 1, 3, &p->capy_act_pos, &p->capy_status_1
, &p->capy_status_2, "Cap Y #9 &5 U", "capy",
lspmac_moveabs_queue);
capz = lspmac_motor_init( &(lspmac_motors
[ 9]), 1, 4, &p->capz_act_pos, &p->capz_status_1
, &p->capz_status_2, "Cap Z #10 &5 V", "capz",
lspmac_moveabs_queue);
scint = lspmac_motor_init( &(lspmac_motors
[10]), 2, 0, &p->scint_act_pos, &p->scint_status_1
, &p->scint_status_2, "Scin Z #11 &5 W", "scint",
lspmac_moveabs_queue);
cenx = lspmac_motor_init( &(lspmac_motors
[11]), 2, 1, &p->centerx_act_pos, &p->centerx_status_1
, &p->centerx_status_2, "Cen X #17 &2 X", "centering.x",
lspmac_moveabs_queue);
ceny = lspmac_motor_init( &(lspmac_motors
[12]), 2, 2, &p->centery_act_pos, &p->centery_status_1
, &p->centery_status_2, "Cen Y #18 &2 Y", "centering.y",
lspmac_moveabs_queue);
kappa = lspmac_motor_init( &(lspmac_motors
[13]), 2, 3, &p->kappa_act_pos, &p->kappa_status_1
, &p->kappa_status_2, "Kappa #19 &7 X", "kappa",
lspmac_moveabs_queue);
phi = lspmac_motor_init( &(lspmac_motors[
14]), 2, 4, &p->phi_act_pos, &p->phi_status_1
, &p->phi_status_2, "Phi #20 &7 Y", "phi",
lspmac_moveabs_queue);

fshut = lspmac_fshut_init( &(lspmac_motors
[15]));
flight = lspmac_dac_init( &(lspmac_motors[1
6]), &p->front_dac, "M1200", "frontLight.intensity",
lspmac_movedac_queue);
blight = lspmac_dac_init( &(lspmac_motors[1
7]), &p->back_dac, "M1201", "backLight.intensity",
lspmac_movedac_queue);
fscint = lspmac_dac_init( &(lspmac_motors[1
8]), &p->scint_piezo, "M1203", "scint.focus",
lspmac_movedac_queue);

```

```

smart_mag_oo = lspmac_bo_init( &(lspmac_motors
    [19]), "smartMagnet", "M1100=%d", &(md2_status.acc11c_5), 0x01)
;
blight_ud = lspmac_bo_init( &(lspmac_motors
    [20]), "backLight", "M1101=%d", &(md2_status.acc11c_5), 0x02)
;
cryo = lspmac_bo_init( &(lspmac_motors
    [21]), "cryo", "M1102=%d", &(md2_status.acc11c_5), 0x04)
;
dryer = lspmac_bo_init( &(lspmac_motors
    [22]), "dryer", "M1103=%d", &(md2_status.acc11c_5), 0x08)
;
fluo = lspmac_bo_init( &(lspmac_motors
    [23]), "fluo", "M1104=%d", &(md2_status.acc11c_5), 0x10)
;
flight_oo = lspmac_soft_motor_init( &(
    lspmac_motors[24]), "frontLight",
    lspmac_moveabs_frontlight_oo_queue);
blight_f = lspmac_soft_motor_init( &(
    lspmac_motors[25]), "backLight.factor",
    lspmac_moveabs_blight_factor_queue);
flight_f = lspmac_soft_motor_init( &(
    lspmac_motors[26]), "frontLight.factor",
    lspmac_moveabs_flight_factor_queue);

lp_air = lspmac_bi_init( &(lspmac_bis[
    0]), &(md2_status.acc11c_1), 0x01, "Low Pressure Air OK", "
    Low Pressure Air Failed");
hp_air = lspmac_bi_init( &(lspmac_bis[
    1]), &(md2_status.acc11c_1), 0x02, "High Pressure Air OK", "
    High Pressure Air Failed");
cryo_switch = lspmac_bi_init( &(lspmac_bis
    [ 2]), &(md2_status.acc11c_1), 0x04, "CryoSwitchChanged",
    "CryoSwitchChanged");
blight_down = lspmac_bi_init( &(lspmac_bis
    [ 3]), &(md2_status.acc11c_1), 0x08, "Backlight Down",
    "Backlight Not Down");
blight_up = lspmac_bi_init( &(lspmac_bis
    [ 4]), &(md2_status.acc11c_1), 0x10, "Backlight Up",
    "Backlight Not Up");
cryo_back = lspmac_bi_init( &(lspmac_bis
    [ 5]), &(md2_status.acc11c_1), 0x40, "Cryo Back",
    "Cryo Not Back");
fluor_back = lspmac_bi_init( &(lspmac_bis
    [ 6]), &(md2_status.acc11c_2), 0x01, "Fluor. Det. Parked",
    "Fluor. Det. Not Parked");
sample_detected = lspmac_bi_init( &(lspmac_bis
    [ 7]), &(md2_status.acc11c_2), 0x02, "SamplePresent",
    "SampleAbsent");
etel_ready = lspmac_bi_init( &(lspmac_bis
    [ 8]), &(md2_status.acc11c_2), 0x20, "ETEL Ready",
    "ETEL Not Ready");
etel_on = lspmac_bi_init( &(lspmac_bis
    [ 9]), &(md2_status.acc11c_2), 0x40, "ETEL On",
    "ETEL Off");
etel_init_ok = lspmac_bi_init( &(lspmac_bis
    [10]), &(md2_status.acc11c_2), 0x80, "ETEL Init OK",
    "ETEL Init Not OK");
minikappa_ok = lspmac_bi_init( &(lspmac_bis
    [11]), &(md2_status.acc11c_3), 0x01, "Minikappa OK",
    "Minikappa Not OK");
smart_mag_on = lspmac_bi_init( &(lspmac_bis
    [12]), &(md2_status.acc11c_3), 0x04, "Smart Magnet On",
    "Smart Magnet Not On");
arm_parked = lspmac_bi_init( &(lspmac_bis
    [13]), &(md2_status.acc11c_3), 0x08, "Arm Parked",
    "Arm Not Parked");
smart_mag_err = lspmac_bi_init( &(lspmac_bis
    [14]), &(md2_status.acc11c_3), 0x10, "Smart Magnet Error",
    "Smart Magnet OK");
shutter_open = lspmac_bi_init( &(lspmac_bis
    [15]), &(md2_status.acc11c_3), 0x100, "Shutter Open",
    "Shutter Not Open");
smart_mag_off = lspmac_bi_init( &(lspmac_bis
    [16]), &(md2_status.acc11c_5), 0x01, "Smart Magnet Off",
    "Smart Magnet Not Off");

//
// Initialize several commands that get called, perhaps, alot
//

rr_cmd.RequestType = VR_UPLOAD;
rr_cmd.Request = VR_PMAC_READREADY;
rr_cmd.wValue = 0;

```



```

rr_cmd.wIndex      = 0;
rr_cmd.wLength     = htons(2);
memset( rr_cmd.bData, 0, sizeof(rr_cmd.bData));

gb_cmd.RequestType = VR_UPLOAD;
gb_cmd.Request     = VR_PMAC_GETBUFFER;
gb_cmd.wValue      = 0;
gb_cmd.wIndex      = 0;
gb_cmd.wLength     = htons(1400);
memset( gb_cmd.bData, 0, sizeof(gb_cmd.bData));

cr_cmd.RequestType = VR_UPLOAD;
cr_cmd.Request     = VR_CTRL_RESPONSE;
cr_cmd.wValue      = 0;
cr_cmd.wIndex      = 0;
cr_cmd.wLength     = htons(1400);
memset( cr_cmd.bData, 0, sizeof(cr_cmd.bData));

//
// Initialize some mutexs and conditions
//

pthread_mutex_init( &pmac_queue_mutex, &mutex_initializer);
pthread_cond_init( &pmac_queue_cond, NULL);

lspmac_shutter_state = 0; //
    assume the shutter is now closed: not a big deal if we are wrong
pthread_mutex_init( &lspmac_shutter_mutex, &
    mutex_initializer);
pthread_cond_init( &lspmac_shutter_cond, NULL);
pmacfd.fd = -1;

pthread_mutex_init( &lspmac_moving_mutex, &
    mutex_initializer);
pthread_cond_init( &lspmac_moving_cond, NULL);

pthread_mutex_init( &lspmac_ascii_mutex, &mutex_initializer
);

pthread_mutex_init( &lspmac_ascii_buffers_mutex, &
    mutex_initializer);

//
// clear the ascii communications buffers
//
{
    uint32_t cc;
    cc = 0;
    lspmac_send_command( VR_UPLOAD, VR_PMAC_SETMEM
        , 0x0e9e, 0, 4, (char *)&cc, NULL, 1, NULL);

    cc = 0x18;
    lspmac_send_command( VR_UPLOAD, VR_PMAC_SETMEM
        , 0x0e9e, 0, 4, (char *)&cc, NULL, 1, NULL);
}

lspmac_SockSendDPLine( NULL, "I5=0");
lspmac_SockSendDPLine( NULL, "ENABLE PLCC 0,2");
lspmac_SockSendDPLine( NULL, "DISABLE PLCC 1");
lspmac_SockSendDPLine( NULL, "I5=3");
}

```

7.6.4.38 int lspmac_jogabs_queue (lspmac_motor_t * mp, double requested_position)

Use jog to move motor to requested position.

Parameters

in	<i>mp</i>	The motor to move
in	<i>requested_position</i>	Where to move it

Definition at line 2871 of file lspmac.c.

```

{

return lspmac_move_or_jog_abs_queue( mp,

```

```

        requested_position, 1);
}

```

7.6.4.39 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 3328 of file lspmac.c.

```

{
    int z;

    z = round(lspmac_getPosition( zoom));

    lslogging_log_message( "lspmac_light_zoom_cb: zoom = %d"
        , z);

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

```

7.6.4.40 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 378 of file lspmac.c.

```

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

    foundone = 0;
    if( lut != NULL && nlut > 1) {
        for( i=0; i < 2*nlut; i += 2) {
            x1 = lut[i];
            y1 = lut[i+1];
            if( i < 2*nlut - 2) {
                x2 = lut[i+2];
                y2 = lut[i+3];
            }

            //
            // First one too big? Use the y value of the first element
            //
            if( i == 0 && x1 > x) {
                y = y1;
            }
        }
    }
}

```

```

        foundone = 1;
        break;
    }

    //
    // Look for equality
    //
    if( x1 == x ) {
        y = y1;
        foundone = 1;
        break;
    }

    //
    // Maybe interpolate
    //
    if( (i < 2*nlut-2) && x < x2) {
        m = (y2 - y1) / (x2 - x1);
        y = m*(x - x1) + y1;
        foundone = 1;
        break;
    }
}
if( foundone == 0) {
    // must be bigger than the last entry
    //
    //
    y = lut[2*(nlut-1) + 1];
}
return y;
}
return 0.0;
}

```

7.6.4.41 void lspmac_more_ascii.cb (pmac_cmd_queue_t * cmd, int nreceived, char * buff)

we are expecting more characters from the DPRAM ASCII interface

Definition at line 1821 of file lspmac.c.

```

{
    lspmac_get_ascii( cmd->event );
}

```

7.6.4.42 lspmac_motor_t* lspmac_motor_init (lspmac_motor_t * d, int wy, int wx, int * posp, int * stat1p, int * stat2p, char * wtitle, char * name, int(*) (lspmac_motor_t *, double) moveAbs)

Initialize a pmac stepper or servo motor.

Parameters

in, out	<i>d</i>	An uninitialized motor object
in	<i>wy</i>	Curses status window row index
in	<i>wx</i>	Curses status window column index
in	<i>posp</i>	Pointer to position status
in	<i>stat1p</i>	Pointer to 1st status word
in	<i>stat2p</i>	Pointer to 2nd status word
in	<i>wtitle</i>	Title for this motor (to display)
in	<i>name</i>	This motor's name
in	<i>moveAbs</i>	Method to use to move this motor

Definition at line 3005 of file lspmac.c.

```

{
    _lspmac_motor_init( d, name );

    d->moveAbs      = moveAbs;
    d->read          = lspmac_pmacmotor_read;
}

```

```

d->actual_pos_cnts_p = posp;
d->status1_p         = stat1p;
d->status2_p         = stat2p;
d->win = newwin( LS_DISPLAY_WINDOW_HEIGHT,
                LS_DISPLAY_WINDOW_WIDTH, wy*LS_DISPLAY_WINDOW_HEIGHT
                , wx*LS_DISPLAY_WINDOW_WIDTH);
box( d->win, 0, 0);
mvwprintw( d->win, 1, 1, "%s", wtitle);
wnoutrefresh( d->win);

return d;
}

```

7.6.4.43 int lsrmac_move_or_jog_abs_queue(lsrmac_motor_t * mp, double requested_position, int use_jog)

Move method for normal stepper and servo motor objects Returns non-zero on abort, zero if OK.

< format string for coordinate system move

< coordinate system bit

< the requested position in units of "counts"

< motor and coordinate system;

< our axis

Parameters

in	<i>mp</i>	The motor to move
in	<i>requested_ - position</i>	Where to move it
in	<i>use_jog</i>	1 to force jog, 0 for motion prog

Definition at line 2664 of file lsrmac.c.

```

{
char *fmt;
int q100;
int requested_pos_cnts;
int coord_num, motor_num;
char *axis;
double u2c;
double neutral_pos;
double min_pos, max_pos;
int pos_limit_hit, neg_limit_hit;
struct timespec timeout, now;
int err;

pthread_mutex_lock( &(mp->mutex));

u2c          = lsredis_getd( mp->u2c);
motor_num    = lsredis_getl( mp->motor_num);
coord_num    = lsredis_getl( mp->coord_num);
axis         = lsredis_getstr( mp->axis);
neutral_pos  = lsredis_getd( mp->neutral_pos);
min_pos      = lsredis_getd( mp->min_pos);
max_pos      = lsredis_getd( mp->max_pos);
pos_limit_hit = lsredis_getd( mp->pos_limit_hit);
neg_limit_hit = lsredis_getd( mp->neg_limit_hit);

if( u2c == 0.0 || requested_position < min_pos || requested_position >
    max_pos) {
    //
    // Shouldn't try moving a motor that's in trouble
    //
    pthread_mutex_unlock( &(mp->mutex));
    lslogging_log_message( "lsrmac_move_or_jog_abs_queue:
        %s u2c=%f requested position=%f min allowed=%f max allowed=%f", mp->name
        , u2c, requested_position, min_pos, max_pos);
    lsevents_send_event( "%s Move Aborted", mp->name);
    return 1;
}

if( (neg_limit_hit && (requested_position < mp->position)) || (pos_limit_hit
    && (requested_position > mp->position))) {
    pthread_mutex_unlock( &(mp->mutex));
}

```

```

lslogging_log_message( "lspmac_move_or_jog_abs_queue:
    %s Moving wrong way on limit: requested position=%f  current position=%f  low
    limit=%d high limit=%d",
        mp->name, requested_position, mp->position
        , neg_limit_hit, pos_limit_hit);
lsevents_send_event( "%s Move Aborted", mp->name);
return 2;
}

mp->requested_position = requested_position;
mp->not_done = 1;
mp->motion_seen = 0;
mp->command_sent = 0;
mp->requested_pos_cnts = u2c * (requested_position +
    neutral_pos);
requested_pos_cnts = mp->requested_pos_cnts;

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

pthread_mutex_unlock( &(mp->mutex));

if( !use_jog) {
    //
    // Make sure the coordinate system is not moving something, wait if it is
    //
    pthread_mutex_lock( &lspmac_moving_mutex);

    clock_gettime( CLOCK_REALTIME, &now);
    //
    // TODO: Have all moves estimate how long they'll take and use that here
    //
    timeout.tv_sec = now.tv_sec + 60.0;          // a long timeout, but
    // we might really be moving something that takes this long (or longer)
    timeout.tv_nsec = now.tv_nsec;

    err = 0;
    while( err == 0 && (lspmac_moving_flags & q100) != 0)
        err = pthread_cond_timedwait( &lspmac_moving_cond, &
            lspmac_moving_mutex, &timeout);

    pthread_mutex_unlock( &lspmac_moving_mutex);

    if( err == ETIMEDOUT) {
        lslogging_log_message( "
            lspmac_move_or_jog_abs_queue: Done.  lspmac_moving_flags = %0x", lspmac_moving_flags);
        lsevents_send_event( "%s Move Aborted", mp->name);
        return 1;
    }

    //
    // Set the "we are moving this coordinate system" flag
    //
    lspmac_SockSendDPLine( NULL, "M5075=(M5075 | %d)",
        q100);

    switch( *axis) {
    case 'A':
        fmt = "%d Q16=%d Q100=%d B146R";
        break;

    case 'B':
        fmt = "%d Q17=%d Q100=%d B147R";
        break;

    case 'C':
        fmt = "%d Q18=%d Q100=%d B148R";
        break;
    case 'X':
        fmt = "%d Q10=%d Q100=%d B140R";
        break;

    case 'Y':
        fmt = "%d Q11=%d Q100=%d B141R";
        break;

    case 'Z':
        fmt = "%d Q12=%d Q100=%d B142R";
        break;

    case 'U':
        fmt = "%d Q13=%d Q100=%d B143R";

```

```

        break;

    case 'V':
        fmt = "%d Q14=%d Q100=%d B144R";
        break;

    case 'W':
        fmt = "%d Q15=%d Q100=%d B145R";
        break;
    }

    //
    // Make sure the flag has been seen
    //

    clock_gettime( CLOCK_REALTIME, &now);
    timeout.tv_sec = now.tv_sec + 4.0; // also a long timeout.
    // This should really only take a few milliseconds on a slow day
    timeout.tv_nsec = now.tv_nsec;

    pthread_mutex_lock( &lspmac_moving_mutex);

    err = 0;
    while( err == 0 && (lspmac_moving_flags & q100) == 0)
        err = pthread_cond_timedwait( &lspmac_moving_cond, &
            lspmac_moving_mutex, &timeout);
    pthread_mutex_unlock( &lspmac_moving_mutex);

    if( err == ETIMEDOUT) {
        lslogging_log_message( "
            lspmac_move_or_jog_abs_queue: Did not see flag propagate. Move aborted. lspmac_moving_flags = %0x",
            lspmac_moving_flags);
        lsevents_send_event( "%s Move Aborted", mp->name);
        return 1;
    }
}

pthread_mutex_lock( &(mp->mutex));
if( use_jog) {
    lspmac_SockSendDPLine( mp->name, "%d j=%d",
        motor_num, requested_pos_cnts);
} else {
    lspmac_SockSendDPLine( mp->name, fmt, coord_num,
        requested_pos_cnts, q100);
}
pthread_mutex_unlock( &(mp->mutex));

free( axis);

return 0;
}

```

7.6.4.44 int lspmac_move_or_jog_preset_queue(lspmac_motor_t * mp, char * preset, int use_jog)

move using a preset value returns 0 on success, non-zero on error

Parameters

in	<i>mp</i>	Our motor
in	<i>preset</i>	the name of the preset
	<i>use_jog</i>	[in] 1 to force jog, 0 to try motion prog

Definition at line 2833 of file lspmac.c.

```

{
    double pos;
    int err;
    int rtn;

    if( preset == NULL || *preset == 0) {
        lsevents_send_event( "%s Move Aborted", mp->name);
        return 0;
    }

    err = lsredis_find_preset( mp->name, preset, &pos);

    if( err != 0)
        rtn = lspmac_move_or_jog_abs_queue( mp, pos,

```

```

        use_jog);
    else {
        lsevents_send_event( "%s Move Aborted", mp->name);
        rtn = 1;
    }
    return rtn;
}

```

7.6.4.45 int lspmac_move_preset_queue (lspmac_motor_t * mp, char * preset_name)

Move a given motor to one of its preset positions.

No movement if the preset is not found.

Parameters

<i>mp</i>	lspmac motor pointer
<i>preset_name</i>	Name of the preset to use

Definition at line 2351 of file lspmac.c.

```

{
    double pos;
    int err;

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

    err = lsredis_find_preset( mp->name, preset_name, &pos
    );
    if( err == 0)
        return 1;

    err = mp->moveAbs( mp, pos);
    if( !err)
        lslogging_log_message( "lspmac_move_preset_queue:
            moving %s to preset '%s' (%f)", mp->name, preset_name, pos);
    //
    // the abort event should have been sent in moveAbs
    //
    return err;
}

```

7.6.4.46 int lspmac_moveabs_blight_factor_queue (lspmac_motor_t * mp, double pos)

Definition at line 2544 of file lspmac.c.

```

{
    char *fmt;

    if( pos >= 60 && pos <= 140) {
        pthread_mutex_lock( &(mp->mutex));
        *mp->actual_pos_cnts_p = pos;
        mp->position = pos;
        pthread_mutex_unlock( &(mp->mutex));

        pthread_mutex_lock( &(blight->mutex));
        fmt = lsredis_getstr( blight->redis_fmt);
        lsredis_setstr( blight->u2c, fmt, pos / 100.0);
        free( fmt);
        pthread_mutex_unlock( &(blight->mutex));

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

    return 0;
}

```

7.6.4.47 int lspmac_moveabs_bo_queue (lspmac_motor_t * mp, double requested_position)

Move method for binary i/o motor objects.

Parameters

in	<i>mp</i>	A binary i/o motor object
in	<i>requested_position</i>	a 1 or a 0 request to move

Definition at line 2425 of file lspmac.c.

```

{

pthread_mutex_lock( &(mp->mutex));
mp->requested_position = requested_position == 0.0 ? 0.0 :
    1.0;
mp->requested_pos_cnts = requested_position == 0.0 ? 0 : 1;

mp->not_done = 1;
mp->motion_seen = 0;
lspmac_SockSendDPLine( mp->name, mp->write_fmt
    , mp->requested_pos_cnts);

pthread_mutex_unlock( &(mp->mutex));
return 0;
}

```

7.6.4.48 int lspmac_moveabs_flight_factor_queue (lspmac_motor_t * mp, double pos)

Definition at line 2521 of file lspmac.c.

```

{

char *fmt;

if( pos >= 60 && pos <= 140) {
pthread_mutex_lock( &(mp->mutex));
*mp->actual_pos_cnts_p = pos;
mp->position = pos;
pthread_mutex_unlock( &(mp->mutex));

pthread_mutex_lock( &(flight->mutex));

fmt = lsredis_getstr( flight->redis_fmt);
lsredis_setstr( flight->u2c, fmt, pos / 100.0);
free( fmt);

pthread_mutex_unlock( &(flight->mutex));

flight->moveAbs( flight, lspmac_getPosition
    ( zoom));
return 0;
}
return 1;
}

```

7.6.4.49 int lspmac_moveabs_frontlight_oo_queue (lspmac_motor_t * mp, double pos)

"move" frontlight on/off

Definition at line 2508 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));
    }
    return 0;
}

```

7.6.4.50 int lspmac_moveabs.fshut_queue (lspmac_motor_t * mp, double requested_position)

Move method for the fast shutter.

Slightly more complicated than a binary io as some flags need to be set up.

Parameters

<i>mp</i>	The fast shutter motor instance
<i>requested_position</i>	1 (open) or 0 (close), really

Definition at line 2395 of file lspmac.c.

```

    {
        pthread_mutex_lock( &(mp->mutex));

        mp->requested_position = requested_position;
        mp->not_done = 1;
        mp->motion_seen = 0;
        mp->requested_pos_cnts = requested_position;
        if( requested_position != 0) {
            //
            // ScanEnable=0, ManualEnable=1, ManualOn=1
            //
            lspmac_SockSendDPLine( mp->name, "M1124=0 M1125=1
                M1126=1");
        } else {
            //
            // ManualOn=0, ManualEnable=0, ScanEnable=0
            //
            lspmac_SockSendDPLine( mp->name, "M1126=0 M1125=0
                M1124=0");
        }

        pthread_mutex_unlock( &(mp->mutex));

        return 0;
    }

```

7.6.4.51 int lspmac_moveabs.queue (lspmac_motor_t * mp, double requested_position)

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

Parameters

in	<i>mp</i>	The motor to move
in	<i>requested_position</i>	Where to move it

Definition at line 2861 of file lspmac.c.

```

    {

        return lspmac_move_or_jog_abs_queue( mp,
            requested_position, 0);
    }

```

7.6.4.52 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 2451 of file lspmac.c.

```

    {
// 240          LS-CAT Timed X move
//          Q10   = Starting X value (cnts)
//          Q11   = Delta X value   (cnts)
//          Q12   = Time to run between the two points (mSec)
//          Q13   = Acceleration time (msecs)
//          Q100  = 1 << (coord sys no - 1)

int q10;      // Starting value (counts)
int q11;      // Delta (counts)
int q12;      // Time to run (msecs)
int q13;      // Acceleration time (msecs)
int q100;     // 1 << (coord sys no - 1)
int coord_num; // our coordinate number
double u2c;
double neutral_pos;
double max_accel;

pthread_mutex_lock( &(mp->mutex));

u2c      = lsredis_getd( mp->u2c);
max_accel = lsredis_getd( mp->max_accel);
coord_num = lsredis_getl( mp->coord_num);
neutral_pos = lsredis_getd( mp->neutral_pos);

if( u2c == 0.0 || time <= 0.0 || max_accel <= 0.0) {
//
// Shouldn't try moving a motor that has bad motion parameters
//
pthread_mutex_unlock( &(mp->mutex));
return;
}

mp->not_done      = 1;
mp->motion_seen   = 0;

mp->requested_position = start + delta;
mp->requested_pos_cnts = u2c * (mp->requested_position
+ neutral_pos);
q10 = mp->requested_pos_cnts;
q11 = u2c * delta;
q12 = 1000 * time;
q13 = q11 / q12 / max_accel;
q100 = 1 << (coord_num - 1);
pthread_mutex_unlock( &(mp->mutex));

pthread_mutex_lock( &(mp->mutex));
lspmac_SockSendDPLine( mp->name, "%d Q10=%d Q11=%d
Q12=%d Q13=%d Q100=%d B240R", coord_num, q10, q11, q12, q13, q100);
pthread_mutex_unlock( &(mp->mutex));
}

```

7.6.4.53 int lspmac_moveabs_wait (lspmac_motor_t * mp, double timeout_secs)

Wait for motor to finish moving.

Assume motion already queued, now just wait

Parameters

<i>mp</i>	The motor object to wait for
<i>timeout_secs</i>	The number of seconds to wait for. Fractional values fine.

Definition at line 2886 of file lspmac.c.

```

{
    struct timespec timeout, now;
    double isecs, fsecs;
    int err;

    //
    // Copy the queue item for the most recent move request
    //
    clock_gettime( CLOCK_REALTIME, &now);

    fsecs = modf( timeout_secs, &isecs);

    timeout.tv_sec = now.tv_sec + (long)floor( isecs);
    timeout.tv_nsec = now.tv_nsec + (long)floor( fsecs * 1.0e9);

    timeout.tv_sec += timeout.tv_nsec / 1000000000;
    timeout.tv_nsec %= 1000000000;

    err = 0;
    pthread_mutex_lock( &(mp->mutex));

    while( err == 0 && mp->command_sent == 0)
        err = pthread_cond_timedwait( &mp->cond, &mp->mutex, &timeout);
    pthread_mutex_unlock( &(mp->mutex));
    if( err != 0) {
        if( err != ETIMEDOUT) {
            lslogging_log_message( "lspmac_moveabs_wait:
                unexpected error from timedwait: %d tv_sec %ld tv_nsec %ld", err, timeout.tv_sec,
                timeout.tv_nsec);
        }
        return 1;
    }

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

    err = 0;
    pthread_mutex_lock( &(mp->mutex));
    while( err == 0 && mp->motion_seen == 0)
        err = pthread_cond_timedwait( &(mp->cond), &(mp->mutex), &timeout)
        ;

    if( err != 0) {
        if( err != ETIMEDOUT) {
            lslogging_log_message( "lspmac_moveabs_wait:
                unexpected error from timedwait: %d tv_sec %ld tv_nsec %ld", err, timeout.tv_sec,
                timeout.tv_nsec);
        }
        pthread_mutex_unlock( &(mp->mutex));
        return 1;
    }

    //
    // wait for the motion that we know has started to finish
    //
    err = 0;
    while( err == 0 && mp->not_done)
        err = pthread_cond_timedwait( &(mp->cond), &(mp->mutex), &timeout)
        ;

    if( err != 0) {
        if( err != ETIMEDOUT) {
            lslogging_log_message( "lspmac_moveabs_wait:
                unexpected error from timedwait: %d tv_sec %ld tv_nsec %ld", err, timeout.tv_sec,
                timeout.tv_nsec);
        }
        pthread_mutex_unlock( &(mp->mutex));
        return 1;
    }

    //
    // if return code was not 0 then we know we shouldn't wait for not_done flag.
    // In this case the motion ended before we read the status registers
    //
    pthread_mutex_unlock( &(mp->mutex));
}

```

```

    return 0;
}

```

7.6.4.54 int lspmac_movedac.queue (lspmac_motor_t * mp, double requested_position)

Move method for dac motor objects (ie, lights)

Parameters

in	<i>mp</i>	Our motor
in	<i>requested_position</i>	Desired x postion (look up and send y position)

Definition at line 2285 of file lspmac.c.

```

{
double u2c;

pthread_mutex_lock( &(mp->mutex));

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

if( mp->nlut > 0 && mp->lut != NULL) {
//
// u2c scales the lookup table value
//
mp->requested_pos_cnts = u2c * lspmac_lut( mp->
nlut, mp->lut, requested_position);

lslogging_log_message( "lspmac_movedac_queue: motor %s
requested position %f requested counts %d u2c %f",
mp->name, mp->requested_position
, mp->requested_pos_cnts, u2c);

mp->not_done = 1;
mp->motion_seen = 0;

lspmac_SockSendDPlane( mp->name, "%s=%d", mp->
dac_mvar, mp->requested_pos_cnts);
}

pthread_mutex_unlock( &(mp->mutex));
return 0;
}

```

7.6.4.55 int lspmac_movezoom.queue (lspmac_motor_t * mp, double requested_position)

Move method for the zoom motor.

Parameters

in	<i>mp</i>	the zoom motor
in	<i>requested_position</i>	our desired zoom

Definition at line 2318 of file lspmac.c.

```

{
double y;
int motor_num;

pthread_mutex_lock( &(mp->mutex));

motor_num = lsredis_getl( mp->motor_num);

mp->requested_position = requested_position;

if( mp->nlut > 0 && mp->lut != NULL) {

```

```

y = lspmac_lut( mp->nlut, mp->lut, requested_position);

mp->requested_pos_cnts = (int)y;
mp->not_done = 1;
mp->motion_seen = 0;

lspmac_SockSendDPline( mp->name, "##d j=%d",
    motor_num, mp->requested_pos_cnts);
}
pthread_mutex_unlock( &(mp->mutex));
return 0;
}

```

7.6.4.56 void lspmac_next_state ()

State machine logic.

Given the current state, generate the next one

Definition at line 2116 of file lspmac.c.

```

{

//
// Connect to the pmac and perhaps initialize it.
// OK, this is slightly more than just the state
// machine logic...
//
if( ls_pmac_state == LS_PMAC_STATE_DETACHED
) {
//
// TODO (eventually)
// This ip address wont change in a single PMAC installation
// We'll need to audit the code if we decide to implement
// multiple PMACs so might as well wait til then.
//
lsConnect( "192.6.94.5");

//
// If the connect was successful we can proceed with the initialization
//
if( ls_pmac_state != LS_PMAC_STATE_DETACHED
) {
lspmac_SockFlush();

//
// Harvest the I and M variables in case we need them
// one day.
//
if( getmvars) {
lspmac_GetAllMVars();
getmvars = 0;
}

if( getivars) {
lspmac_GetAllIVars();
getivars = 0;
}
}
}

//
// Check the command queue and perhaps go to the "Send Command" state.
//
if( ls_pmac_state == LS_PMAC_STATE_IDLE) {
int goodtogo;
goodtogo = 0;
pthread_mutex_lock( &lspmac_ascii_mutex);
if( lspmac_ascii_busy==0 && lspmac_dpascii_on
!= lspmac_dpascii_off)
goodtogo = 1;
pthread_mutex_unlock( &lspmac_ascii_mutex);
if( goodtogo)
lspmac_SockSendDPqueue();
}

if( ls_pmac_state == LS_PMAC_STATE_IDLE &&
ethCmdOn != ethCmdOff)
ls_pmac_state = LS_PMAC_STATE_SC;
}

```

```

//
// Set the events flag
// to tell poll what we are waiting for.
//
switch( ls_pmac_state) {
case LS_PMAC_STATE_DETACHED:
//
// there shouldn't be a valid fd, so ignore the events
//
pmacfd.events = 0;
break;

case LS_PMAC_STATE_IDLE:
if( ethCmdOn == ethCmdOff) {
//
// Anytime we are idle we want to
// get the status of the PMAC
//

lspmac_get_status();
}

//
// These states require that we listen for packets
//
case LS_PMAC_STATE_WACK_NFR:
case LS_PMAC_STATE_WACK:
case LS_PMAC_STATE_WACK_CC:
case LS_PMAC_STATE_WACK_RR:
case LS_PMAC_STATE_WCR:
case LS_PMAC_STATE_WGB:
case LS_PMAC_STATE_GMR:
pmacfd.events = POLLIN;
break;

//
// These states require that we send packets out.
//
case LS_PMAC_STATE_SC:
case LS_PMAC_STATE_CR:
case LS_PMAC_STATE_RR:
case LS_PMAC_STATE_GB:
//
// Sad fact: PMAC will fail to process commands if we send them too
// quickly.
// We deal with that by waiting a tad before we let poll tell us the PMAC
// socket is ready to write.
//
gettimeofday( &now, NULL);
if( ((now.tv_sec * 1000000. + now.tv_usec) - (pmac_time_sent.tv_sec
* 1000000. + pmac_time_sent.tv_usec)) < PMAC_MIN_CMD_TIME) {
pmacfd.events = 0;
} else {
pmacfd.events = POLLOUT;
}
break;
}
}

```

7.6.4.57 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 1350 of file lspmac.c.

```

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

```

```

char *fmt;
int status_changed;

pthread_mutex_lock( &(mp->mutex));

//
// if this time and last time were both "in position"
// and the position changed significantly then log the event
//
// On E omega has been observed to change by 0x10000 on its own
// with no real motion.
//
if( mp->status2 & 1 && mp->status2 == *mp->status2_p
    && abs( mp->actual_pos_cnts - *mp->actual_pos_cnts_p
    ) > 256) {
    // lslogging_log_message( "Instantaneous change: %s old status1: %0x,
    // new status1: %0x, old status2: %0x, new status2: %0x, old cnts: %0x, 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");
}

// Get some values we might need later
//
u2c = lsredis_getd( mp->u2c);
motor_num = lsredis_getl( mp->motor_num);
neutral_pos = lsredis_getd( mp->neutral_pos);

//
// maybe look for omega zero crossing
//
if( motor_num == 1 && omega_zero_search && *mp->
    actual_pos_cnts_p >= 0 && mp->actual_pos_cnts <
    0) {
    int secs, nsecs;

    if( omega_zero_velocity > 0.0) {
        secs = *mp->actual_pos_cnts_p / omega_zero_velocity
        ;
        nsecs = (*mp->actual_pos_cnts_p / omega_zero_velocity
            - secs) * 1000000000;

        omega_zero_time.tv_sec = lspmac_status_time
        .tv_sec - secs;
        omega_zero_time.tv_nsec = lspmac_status_time
        .tv_nsec;
        if( omega_zero_time.tv_nsec < nsecs) {
            omega_zero_time.tv_sec -= 1;
            omega_zero_time.tv_nsec += 1000000000;
        }
        omega_zero_time.tv_nsec -= nsecs;

        lsevents_send_event( "omega crossed zero");
        lslogging_log_message("lspmac_pmacmotor_read: omega
            zero secs %d nsecs %d ozt.tv_sec %ld ozt.tv_nsec %ld, motor cnts %d",

```

```

        secs, nsecs, omega_zero_time.tv_sec,
        omega_zero_time.tv_nsec, *mp->actual_pos_cnts_p
    );
}
omega_zero_search = 0;
}

// Make local copies so we can inspect them in other threads
// without having to grab the status mutex
//
if( mp->status1 != *mp->status1_p || mp->status2 != *
    mp->status2_p ) {
    mp->status1 = *mp->status1_p;
    mp->status2 = *mp->status2_p;
    status_changed = 1;
} else {
    status_changed = 0;
}
mp->actual_pos_cnts = *mp->actual_pos_cnts_p;

//
// See if we are done moving, ie, in position
//
if( mp->status2 & 0x000001 ) {
    if( mp->not_done ) {
        mp->not_done = 0;
        pthread_cond_signal( &(mp->cond));
    }
} else if( mp->not_done == 0 ) {
    mp->not_done = 1;
}

// See if the motor is moving
//
//           move timer           homing
//           123456             123456
if( mp->status1 & 0x020000 || mp->status1 & 0x000400 ) {
    if( mp->motion_seen == 0 ) {
        mp->motion_seen = 1;
        pthread_cond_signal( &(mp->cond));
    }
}

mvwprintw( mp->win, 2, 1, "%s", LS_DISPLAY_WINDOW_WIDTH
    -2, " ");
mvwprintw( mp->win, 2, 1, "%d cts", LS_DISPLAY_WINDOW_WIDTH
    -6, mp->actual_pos_cnts);
mvwprintw( mp->win, 3, 1, "%s", LS_DISPLAY_WINDOW_WIDTH
    -2, " ");

if( mp->nlut > 0 && mp->lut != NULL ) {
    mp->position = lspmac_rlut( mp->nlut, mp->lut, mp
        ->actual_pos_cnts);
} else {
    if( u2c != 0.0 ) {
        mp->position = ((mp->actual_pos_cnts / u2c) -
            neutral_pos);
    } else {
        mp->position = mp->actual_pos_cnts;
    }
}

if( status_changed || fabs(mp->reported_position - mp->
    position) >= lsredis_getd(mp->update_resolution
    )) {
    fmt = lsredis_getstr(mp->redis_fmt);
    lsredis_setstr( mp->redis_position, fmt, mp->
        position);
    free(fmt);
    mp->reported_position = mp->position;
}

fmt = lsredis_getstr( mp->printf_fmt);
snprintf( s, sizeof(s)-1, fmt, 8, mp->position);
s[sizeof(s)-1] = 0;
free( fmt);

//
// indicate limit problems
//
lsredis_setstr( mp->pos_limit_hit, mp->status1
    & 0x200000 ? "1" : "0");
lsredis_setstr( mp->neg_limit_hit, mp->status1
    & 0x400000 ? "1" : "0");

```



```

// set flag if we are not homed
homing1 = 0;
// ~ (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;
    lsevents_send_event( "%s Homed", mp->name);
}

s[sizeof(s)-1] = 0;
mvwprintw( mp->win, 3, 1, "%*s", LS_DISPLAY_WINDOW_WIDTH
-6, s);

if( status_changed) {
    mvwprintw( mp->win, 4, 1, "%*x", LS_DISPLAY_WINDOW_WIDTH
-2, mp->status1);
    mvwprintw( mp->win, 5, 1, "%*x", LS_DISPLAY_WINDOW_WIDTH
-2, mp->status2);
    sp = "";
    if( mp->status2 & 0x000002)
        sp = "Following Warning";
    else if( mp->status2 & 0x000004)
        sp = "Following Error";
    else if( mp->status2 & 0x000020)
        sp = "I2T Amp Fault";
    else if( mp->status2 & 0x000008)
        sp = "Amp. Fault";
    else if( mp->status2 & 0x000800)
        sp = "Stopped on Limit";
    else if( mp->status1 & 0x040000)
        sp = "Open Loop";
    else if( ~(mp->status1) & 0x080000)
        sp = "Motor Disabled";
    else if( mp->status1 & 0x000400)
        sp = "Homing";
    else if( (mp->status1 & 0x600000) == 0x600000)
        sp = "Both Limits Tripped";
    else if( mp->status1 & 0x200000)
        sp = "Positive Limit";
    else if( mp->status1 & 0x400000)
        sp = "Negative Limit";
    else if( ~(mp->status2) & 0x000400)
        sp = "Not Homed";
    else if( mp->status1 & 0x020000)
        sp = "Moving";
    else if( mp->status2 & 0x000001)
        sp = "In Position";

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

    lsredis_setstr( mp->status_str, sp);
}
wnoutrefresh( mp->win);

pthread_mutex_unlock( &(mp->mutex));

if( homing1)
    lspmac_homed_queue( mp);

if( homing2)
    lspmac_home2_queue( mp);

lspmac_status_last_time.tv_sec = lspmac_status_time
.tv_sec;
lspmac_status_last_time.tv_nsec = lspmac_status_time
.tv_nsec;
}

```

7.6.4.58 `pmac_cmd_queue_t*` `lspmac_pop_queue ()`

Remove the oldest queue item.

Used to send command to PMAC. Note that there is a separate reply index to ensure we've know to what command a reply is referring. Returns the item.

Definition at line 652 of file `lspmac.c`.

```

{
    pmac_cmd_queue_t *rtn;

    pthread_mutex_lock( &pmac_queue_mutex);

    if( ethCmdOn == ethCmdOff)
        rtn = NULL;
    else {
        rtn = &(ethCmdQueue[(ethCmdOff++) %
            PMAC_CMD_QUEUE_LENGTH]);
        clock_gettime( CLOCK_REALTIME, &(rtn->time_sent));
    }
    pthread_mutex_unlock( &pmac_queue_mutex);
    return rtn;
}

```

7.6.4.59 `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 672 of file `lspmac.c`.

```

{
    pmac_cmd_queue_t *rtn;

    pthread_mutex_lock( &pmac_queue_mutex);

    if( ethCmdOn == ethCmdReply)
        rtn = NULL;
    else
        rtn = &(ethCmdQueue[(ethCmdReply++) %
            PMAC_CMD_QUEUE_LENGTH]);

    pthread_mutex_unlock( &pmac_queue_mutex);
    return rtn;
}

```

7.6.4.60 `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 628 of file `lspmac.c`.

```

{
    pmac_cmd_queue_t *rtn;

    pthread_mutex_lock( &pmac_queue_mutex);
    rtn = &(ethCmdQueue[(ethCmdOn++) % PMAC_CMD_QUEUE_LENGTH
    ]);
    memcpy( rtn, cmd, sizeof( pmac_cmd_queue_t));
    rtn->time_sent.tv_sec = 0;
    rtn->time_sent.tv_nsec = 0;
    pthread_cond_signal( &pmac_queue_cond);
}

```

```

pthread_mutex_unlock( &pmac_queue_mutex);

return rtn;
}

```

7.6.4.61 void lspmac_Reset ()

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

Definition at line 751 of file lspmac.c.

```

{
ls_pmac_state = LS_PMAC_STATE_IDLE;

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

lspmac_SockFlush();
}

```

7.6.4.62 void lspmac_reset_queue ()

Clear the queue as part of PMAC reinitialization.

Definition at line 615 of file lspmac.c.

```

{
pthread_mutex_lock( &pmac_queue_mutex);
ethCmdOn   = 0;
ethCmdOff  = 0;
ethCmdReply = 0;
pthread_mutex_unlock( &pmac_queue_mutex);
}

```

7.6.4.63 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 436 of file lspmac.c.

```

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

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

    //
    // are the table values going up or down?
    //
    if( lut[1] < lut[2*nlut-1])
        up = 1;
    else
        up = 0;

    //
    // Linear search
    //
    for( i=0; i < 2*nlut; i += 2) {
        x1 = lut[i];
        y1 = lut[i+1];
        if( i < 2*nlut - 2) {
            x2 = lut[i+2];

```

```

        y2 = lut[i+3];
    }
    //
    // see if y is before the beginning of the table
    //
    if( i==0 && ( up ? y1 > y : y1 < y) ) {
        x = x1;
        foundone = 1;
        break;
    }
    //
    // Did we, perhaps, nail it?
    //
    if( y1 == y ) {
        x = x1;
        foundone = 1;
        break;
    }

    //
    // Interpolate between the two values (if we've not bumped our heads on
    // the end of the table)
    //
    if( (i < 2*nlut-2) && (up ? y < y2 : y > y2) ) {
        m = (x2 - x1) / (y2 - y1);
        x = m * (y - y1) + x1;
        foundone = 1;
        break;
    }
}
//
// y is off the charts: just use the last value
//
if( foundone == 0 ) {
    x = lut[2*(nlut-1)];
}
return x;
}
return 0.0;
}

```

7.6.4.64 void lspmac_run ()

Start up the lspmac thread.

Definition at line 3511 of file lspmac.c.

```

{
    char **inits;
    lspmac_motor_t *mp;
    char evts[64];
    int i;
    int active;

    pthread_create( &pmac_thread, NULL, lspmac_worker,
        NULL);

    lsevents_add_listener( "CryoSwitchChanged",
        lspmac_cryoSwitchChanged_cb);
    lsevents_add_listener( "scint In Position",
        lspmac_scint_inPosition_cb);
    lsevents_add_listener( "scintDried",
        lspmac_scint_dried_cb);
    lsevents_add_listener( "backLight 1",
        lspmac_backLight_up_cb);
    lsevents_add_listener( "backLight 0",
        lspmac_backLight_down_cb);
    lsevents_add_listener( "cam.zoom In Position",
        lspmac_light_zoom_cb);

    for( i=0; i<lspmac_nmotors; i++) {
        snprintf( evts, sizeof( evts)-1, "%s command accepted", lspmac_motors
            [i].name);
        evts[sizeof( evts)-1] = 0;
        lsevents_add_listener( evts, lspmac_command_done_cb
            );
    }

    lspmac_zoom_lut_setup();
    lspmac_flight_lut_setup();
}

```

```

lspmac_blight_lut_setup();
lspmac_fscint_lut_setup();

//
// Clear the command interfaces
//
lspmac_SockSendControlCharPrint( NULL, '\x18'
    ;
{
    uint32_t cc;
    cc = 0;
    lspmac_send_command( VR_UPLOAD, VR_PMAC_SETMEM
        , 0x0e9e, 0, 4, (char *)&cc, NULL, 1, NULL);

    cc = 0x18;
    lspmac_send_command( VR_UPLOAD, VR_PMAC_SETMEM
        , 0x0e9e, 0, 4, (char *)&cc, NULL, 1, NULL);
}

//
// Initialize the MD2 pmac (ie, turn on the right plcc's etc)
//
/*
for( inits = lsredis_get_string_array(lspmac_md2_init); *inits != NULL;
    inits++) {
    lspmac_SockSendDPline( NULL, *inits);
}
*/

//
// Initialize the pmac's support for each motor
// (ie, set the various flag for when a motor is active or not)
//
for( i=0; i<lspmac_nmotors; i++) {
    mp = &(lspmac_motors[i]);
    active = lsredis_getb( mp->active);

    // if there is a problem with "active" then don't do anything
    // On the other hand, various combinations of yes/no true/fals 1/0 should
    // work
    //
    switch( active) {
    case 1:
        inits = lsredis_get_string_array( mp->active_init
        );
        break;

    case 0:
        inits = lsredis_get_string_array( mp->
        inactive_init);
        break;

    default:
        lslogging_log_message( "lspmac_run: motor %s is
        neither active nor inactive (!?)", mp->name);
        inits = NULL;
    }
    if( inits != NULL) {
        while( *inits != NULL) {
            lspmac_SockSendDPline( NULL, *inits);
            inits++;
        }
    }
}
}

```

7.6.4.65 void lspmac_scint_dried.cb (char * event)

Turn off the dryer.

Parameters

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

Definition at line 3351 of file lspmac.c.

```

{
lslogging_log_message( "lspmac_scint_dried.cb: Stopping
dryer");

```

```

    dryer->moveAbs( dryer, 0.0);
}

```

7.6.4.66 void lspmac_scint_inPosition.cb (char * event)

Maybe start drying off the scintillator.

Parameters

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

Definition at line 3289 of file lspmac.c.

```

{
    double pos;
    double cover;
    int err;

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

```

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

Compose a packet and send it to the PMAC.

This is the meat of the PMAC communications routines. The queued command is returned.

Parameters

in	<i>rqType</i>	VR_UPLOAD or VR_DOWNLOAD
in	<i>rq</i>	PMAC command (see PMAC User Manual)
in	<i>wValue</i>	Command argument 1
in	<i>wIndex</i>	Command argument 2
in	<i>wLength</i>	Length of data array
in	<i>data</i>	Data array (or NULL)
in	<i>responseCB</i>	Function to call when a response is read from the PMAC
in	<i>no_reply</i>	Flag, non-zero means no reply is expected
in	<i>event</i>	base name for events

Definition at line 690 of file lspmac.c.

```

{
    static pmac_cmd_queue_t cmd;

    cmd.pcmd.RequestType = rqType;
    cmd.pcmd.Request      = rq;
    cmd.pcmd.wValue       = htons(wValue);
    cmd.pcmd.wIndex       = htons(wIndex);
}

```

```

cmd.pcmd.wLength      = htons(wLength);
cmd.onResponse        = responseCB;
cmd.no_reply          = no_reply;
cmd.event             = event;

//
// Setting the message buff bData requires a bit more care to avoid over
// filling it
// or sending garbage in the unused bytes.
//

if( wLength > sizeof( cmd.pcmd.bData)) {
    //
    // Bad things happen if we do not catch this case.
    //
    lslogging_log_message( "Message Length %d longer than
        maximum of %ld, aborting", wLength, sizeof( cmd.pcmd.bData));
    exit( -1);
}
if( data == NULL) {
    memset( cmd.pcmd.bData, 0, sizeof( cmd.pcmd.bData));
} else {
    //
    // This could leave bData non-null terminated. I do not know if this is a
    // problem.
    //
    if( wLength > 0)
        memcpy( cmd.pcmd.bData, data, wLength);
    if( wLength < sizeof( cmd.pcmd.bData))
        memset( cmd.pcmd.bData + wLength, 0, sizeof( cmd.pcmd.bData
            ) - wLength);
}

return lspmac_push_queue( &cmd);
}

```

7.6.4.68 void lspmac_sendcmd (char * *event*, void(*)(*pmac_cmd_queue_t* *, int, char *) *responseCB*, char * *fmt*, ...)

PMAC command with call back.

Parameters

in	<i>event</i>	base name for events
in	<i>responseCB</i>	our callback routine
in	<i>fmt</i>	printf style format string

Definition at line 2095 of file lspmac.c.

```

{
    static char tmps[1024];
    va_list arg_ptr;

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

    lspmac_send_command( VR_DOWNLOAD,
        VR_PMAC_SENDLINE, 0, 0, strlen(tmps), tmps, responseCB, 0,
        event);
}

```

7.6.4.69 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 2076 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, NULL, 0, NULL);
    }

```

7.6.4.70 void lspmac.SendControlReplyPrintCB (pmac_cmd_queue_t * cmd, int nreceived, char * buff)

Receive a reply to a control character Print a "printable" version of the character to the terminal Followed by a hex dump of the response.

Parameters

in	<i>cmd</i>	Queue item this is a reply to
in	<i>nreceived</i>	Number of bytes received
in	<i>buff</i>	Buffer of bytes received

Definition at line 1015 of file lspmac.c.

```

    {
        pthread_mutex_lock( &ncurses_mutex);
        wprintw( term_output, "control-%c: ", '@'+ ntohs(cmd->pcmd.
            wValue));
        pthread_mutex_unlock( &ncurses_mutex);
        hex_dump( nreceived, (unsigned char *)buff);
        pthread_mutex_lock( &ncurses_mutex);
        wnoutrefresh( term_output);
        wnoutrefresh( term_input);
        douupdate();
        pthread_mutex_unlock( &ncurses_mutex);
    }

```

7.6.4.71 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 798 of file lspmac.c.

```

    {
        static char *receiveBuffer = NULL;    // the buffer inwhich to stick our
            incoming characters
        static int receiveBufferSize = 0;      // size of receiveBuffer
        static int receiveBufferIn = 0;        // next location to write to in
            receiveBuffer
        pmac_cmd_queue_t *cmd;                // maybe the
            command we are servicing
        ssize_t nsent, nread;                  // nbytes dealt with
        int i;                                // loop counter
        int foundEOCR;                         // end of command response flag

        if( evt->revents & (POLLERR | POLLHUP | POLLNVAL)) {
            if( evt->fd != -1) {
                close( evt->fd);
                evt->fd = -1;
            }
            ls_pmac_state = LS_PMAC_STATE_DETACHED;
        }
    }

```



```

    return;
}

if( evt->revents & POLLOUT) {

    switch( ls_pmac_state) {
    case LS_PMAC_STATE_DETACHED:
        break;
    case LS_PMAC_STATE_IDLE:
        break;

    case LS_PMAC_STATE_SC:
        cmd = lspmac_pop_queue();
        if( cmd == NULL)
            return;

        if( cmd->pcmd.Request == VR_PMAC_GETMEM) {
            nsent = send( evt->fd, cmd, pmac_cmd_size, 0);
            if( nsent != pmac_cmd_size) {
                lslogging_log_message( "Could only send %d of %d
bytes....Not good.", (int)nsent, (int)(pmac_cmd_size));
            }
        } else {
            nsent = send( evt->fd, cmd, pmac_cmd_size + ntohs(cmd->
pcmd.wLength), 0);
            gettimeofday( &pmac_time_sent, NULL);
            if( nsent != pmac_cmd_size + ntohs(cmd->pcmd.wLength
)) {
                lslogging_log_message( "Could only send %d of %d
bytes....Not good.", (int)nsent, (int)(pmac_cmd_size + ntohs(cmd->
pcmd.wLength)));
            }
        }

        if( cmd->pcmd.Request == VR_PMAC_SENDCTRLCHAR
)
            ls_pmac_state = LS_PMAC_STATE_WACK_CC
;
        else if( cmd->pcmd.Request == VR_PMAC_GETMEM)
            ls_pmac_state = LS_PMAC_STATE_GMR;
        else if( cmd->no_reply == 0)
            ls_pmac_state = LS_PMAC_STATE_WACK;
        else
            ls_pmac_state = LS_PMAC_STATE_WACK_NFR
;
        break;

    case LS_PMAC_STATE_CR:
        nsent = send( evt->fd, &cr_cmd, pmac_cmd_size, 0);
        gettimeofday( &pmac_time_sent, NULL);
        ls_pmac_state = LS_PMAC_STATE_WCR;
        break;

    case LS_PMAC_STATE_RR:
        nsent = send( evt->fd, &rr_cmd, pmac_cmd_size, 0);
        gettimeofday( &pmac_time_sent, NULL);
        ls_pmac_state = LS_PMAC_STATE_WACK_RR;
        break;

    case LS_PMAC_STATE_GB:
        nsent = send( evt->fd, &gb_cmd, pmac_cmd_size, 0);
        gettimeofday( &pmac_time_sent, NULL);
        ls_pmac_state = LS_PMAC_STATE_WGB;
        break;
    }
}

if( evt->revents & POLLIN) {

    if( receiveBufferSize - receiveBufferIn < 1400) {
        char *newbuff;

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

```

```

nread = read( evt->fd, receiveBuffer + receiveBufferIn, 1400);

foundEOCR = 0;
if( ls_pmac_state == LS_PMAC_STATE_GMR) {
    //
    // get memory returns binary stuff, don't try to parse it
    //
    receiveBufferIn += nread;
} else {
    //
    // other commands end in 6 if OK, 7 if not
    //
    for( i=receiveBufferIn; i<receiveBufferIn+nread; i++) {
        if( receiveBuffer[i] == 7) {
            //
            // Error condition
            //
            lspmac_Error( &(receiveBuffer[i]));
            receiveBufferIn = 0;
            return;
        }
        if( receiveBuffer[i] == 6) {
            //
            // End of command response
            //
            foundEOCR = 1;
            receiveBuffer[i] = 0;
            break;
        }
    }
    receiveBufferIn = i;
}

cmd = NULL;

switch( ls_pmac_state) {
case LS_PMAC_STATE_WACK_NFR:
    receiveBuffer[--receiveBufferIn] = 0;
    cmd = lspmac_pop_reply();
    ls_pmac_state = LS_PMAC_STATE_IDLE;
    break;
case LS_PMAC_STATE_WACK:
    receiveBuffer[--receiveBufferIn] = 0;
    ls_pmac_state = LS_PMAC_STATE_RR;
    break;
case LS_PMAC_STATE_WACK_CC:
    receiveBuffer[--receiveBufferIn] = 0;
    ls_pmac_state = LS_PMAC_STATE_CR;
    break;
case LS_PMAC_STATE_WACK_RR:
    receiveBufferIn -= 2;
    if( receiveBuffer[receiveBufferIn])
        ls_pmac_state = LS_PMAC_STATE_GB;
    else
        ls_pmac_state = LS_PMAC_STATE_RR;
    receiveBuffer[receiveBufferIn] = 0;
    break;
case LS_PMAC_STATE_GMR:
    cmd = lspmac_pop_reply();
    ls_pmac_state = LS_PMAC_STATE_IDLE;
    break;
case LS_PMAC_STATE_WCR:
    cmd = lspmac_pop_reply();
    ls_pmac_state = LS_PMAC_STATE_IDLE;
    break;
case LS_PMAC_STATE_WGB:
    if( foundEOCR) {
        cmd = lspmac_pop_reply();
        ls_pmac_state = LS_PMAC_STATE_IDLE;
    } else {
        ls_pmac_state = LS_PMAC_STATE_RR;
    }
    break;
}

if( cmd != NULL && cmd->onResponse != NULL) {
    cmd->onResponse( cmd, receiveBufferIn, receiveBuffer);
    receiveBufferIn = 0;
}
}
}

```

7.6.4.72 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 1171 of file lspmac.c.

```

    {
        //
        // track the shutter state and signal if it has changed
        //
        pthread_mutex_lock( &lspmac_shutter_mutex);
        if( md2_status.fs_has_opened && !
            lspmac_shutter_has_opened && !md2_status.
            fs_is_open) {
            //
            // Here the shutter opened and closed again before we got the memo
            // Treat it as a shutter closed event
            //
            pthread_cond_signal( &lspmac_shutter_cond);
        }
        lspmac_shutter_has_opened = md2_status.
            fs_has_opened;

        if( lspmac_shutter_state != md2_status.
            fs_is_open) {
            lspmac_shutter_state = md2_status.fs_is_open
            ;
            pthread_cond_signal( &lspmac_shutter_cond);
        }

        if( md2_status.fs_is_open) {
            mvwprintw( term_status2, 1, 1, "Shutter Open ");
            mp->position = 1;
        } else {
            mvwprintw( term_status2, 1, 1, "Shutter Closed");
            mp->position = 0;
        }

        pthread_mutex_unlock( &lspmac_shutter_mutex);
    }

```

7.6.4.73 void lspmac_SockFlush ()

Reset the PMAC socket from the PMAC side.

Puts the PMAC into a known communications state

Definition at line 744 of file lspmac.c.

```

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

```

7.6.4.74 pmac_cmd_queue_t* lspmac_SockGetmem (int offset, int nbytes)

Request a chunk of memory to be returned.

Parameters

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

Definition at line 1050 of file lspmac.c.

```

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

```

7.6.4.75 pmac_cmd_queue_t* lspmac_SockSendControlCharPrint (char * event, char c)

Send a control character.

Parameters

in	<i>event</i>	base name for events
	<i>c</i>	The control character to send

Definition at line 1103 of file lspmac.c.

```

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

```

7.6.4.76 void lspmac_SockSendDPControlChar (char * event, char c)

use dpram ascii interface to send a control character

Definition at line 1976 of file lspmac.c.

```

    {
uint16_t buff;

buff = 0x07 & c;
lspmac_send_command( VR_UPLOAD, VR_PMAC_SETMEM
    , 0x0e9e, 0, 2, (char *)&buff, lspmac_SockSendDPControlCharCB
    , 1, event);
lsevents_send_event( "%s queued", event);
    }

```

7.6.4.77 void lspmac_SockSendDPControlCharCB (pmac_cmd_queue_t * cmd, int nreceived, char * buf)

Definition at line 1969 of file lspmac.c.

```

    {
if( cmd->event != NULL && *(cmd->event))
    lsevents_send_event( "%s accepted", cmd->event);
    }

```

7.6.4.78 void lspmac_SockSendDPLine (char * event, char * fmt, ...)

prepare (queue up) a line to send the dpram ascii command interface

Definition at line 1949 of file lspmac.c.

```

    {
va_list arg_ptr;
uint32_t index;
char *p1;

```

```

pthread_mutex_lock( &lspmac_ascii_mutex);
index = lspmac_dpascii_on++ % LSPMAC_DPASCII_QUEUE_LENGTH
;

pl = lspmac_dpascii_queue[index].pl;

va_start( arg_ptr, fmt);
vsprintf( pl, 159, fmt, arg_ptr);
pl[159] = 0;
va_end( arg_ptr);

lspmac_dpascii_queue[index].event = event;

pthread_mutex_unlock( &lspmac_ascii_mutex);
}

```

7.6.4.79 void lspmac.SockSendDPqueue ()

Definition at line 1985 of file lspmac.c.

```

{
lspmac_dpascii_queue_t *qp;
uint32_t mask;
uint32_t clrddata;

pthread_mutex_lock( &lspmac_ascii_mutex);
qp = &(lspmac_dpascii_queue[(lspmac_dpascii_off
++) % LSPMAC_DPASCII_QUEUE_LENGTH]);
lspmac_ascii_busy = 1;
pthread_mutex_unlock( &lspmac_ascii_mutex);

lslogging_log_message( "lspmac_SockSendDPqueue: %s", qp
->p1);

clrddata = 0;          // set the control word to zero
lspmac_send_command( VR_UPLOAD, VR_PMAC_SETMEM
, 0x0f40, 0, 4, (char *)&clrddata, NULL, 1, NULL);
lspmac_send_command( VR_UPLOAD, VR_PMAC_SETMEM
, 0x0e9c, 0, 4, (char *)&clrddata, NULL, 1, NULL);

lspmac_send_command( VR_UPLOAD, VR_PMAC_SETMEM
, 0x0ea0, 0, strlen(qp->p1)+1, qp->p1, NULL, 1, NULL);

mask = 0x0001;
lspmac_send_command( VR_UPLOAD, VR_PMAC_SETBIT
, 0x0e9c, 1, sizeof( mask), (char *)&mask, lspmac_asciicmdCB, 1,
qp->event);

if( qp->event != NULL && *(qp->event) != 0)
    lsevents_send_event( "%s queued", qp->event);
}

```

7.6.4.80 pmac_cmd_queue_t* lspmac.SockSendline (char * event, char * fmt, ...)

Send a one line command.

Uses printf style arguments.

Parameters

in	<i>event</i>	base name for events
in	<i>fmt</i>	Printf style format string

Definition at line 1060 of file lspmac.c.

```

{
va_list arg_ptr;
char payload[1400];

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

```

```

lslogging_log_message( payload);

return lspmac_send_command( VR_DOWNLOAD,
    VR_PMAC_SENDLINE, 0, 0, strlen( payload), payload,
    lspmac_GetShortReplyCB, 0, event);
}

```

7.6.4.81 `lspmac_cmd_queue_t*` `lspmac_sock_sendline_nr` (`char * event`, `char * fmt`, ...)

Send a command and ignore the response.

Parameters

in	<i>event</i>	base name for events
in	<i>fmt</i>	Printf style format string

Definition at line 1083 of file `lspmac.c`.

```

{
    va_list arg_ptr;
    char s[512];

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

    lslogging_log_message( s);

    return lspmac_send_command( VR_DOWNLOAD,
        VR_PMAC_SENDLINE, 0, 0, strlen( s), s, NULL, 1, event);
}

```

7.6.4.82 `lspmac_motor_t*` `lspmac_soft_motor_init` (`lspmac_motor_t * d`, `char * name`, `int(*)`(`lspmac_motor_t *`, `double`) `moveAbs`)

Definition at line 3101 of file `lspmac.c`.

```

{
    _lspmac_motor_init( d, name);

    d->moveAbs = moveAbs;
    d->read = lspmac_soft_motor_read;
    d->actual_pos_cnts_p = calloc( sizeof(int), 1);
    *d->actual_pos_cnts_p = 0;

    return d;
}

```

7.6.4.83 `void` `lspmac_soft_motor_read` (`lspmac_motor_t * p`)

Dummy routine to read a soft motor.

Definition at line 3096 of file `lspmac.c`.

```

{
}

```

7.6.4.84 `int lspmac_test_preset (lspmac_motor_t * mp, char * preset_name, double tolerance)`

see if the motor is within tolerance of the preset 1 means yes, it is 0 mean no it isn't or that the preset was not found
Definition at line 2374 of file lspmac.c.

```

    {
double preset_position;
int err;

err = lsredis_find_preset( mp->name, preset_name, &
    preset_position);
if( err == 0)
    return 0;

if( fabs( preset_position - lspmac_getPosition( mp)) <=
    tolerance)
    return 1;

    return 0;
}

```

7.6.4.85 `void lspmac_video_rotate (double secs)`

Special motion program to collect centering video.

Definition at line 2568 of file lspmac.c.

```

                                {
double q10;                // starting position (counts)
double q11;                // delta counts
double q12;                // milliseconds to run over delta

double u2c;
double neutral_pos;

if( secs <= 0.0)
    return;

omega_zero_search = 1;

pthread_mutex_lock( &(omega->mutex));
u2c = lsredis_getd( omega->u2c);
neutral_pos = lsredis_getd( omega->neutral_pos);

q10 = neutral_pos * u2c;
q11 = 360.0 * u2c;
q12 = 1000 * secs;

omega_zero_velocity = 360.0 * u2c / secs; //
    counts/second to back calculate zero crossing time

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

```

7.6.4.86 `void* lspmac_worker (void * dummy)`

Our lspmac worker thread.

Parameters

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

Definition at line 2234 of file lspmac.c.

```

    {
static int disconnected_notify = 0;
static int old_state;

```

```

old_state = ls_pmac_state;
while( 1 ) {
    int pollrtn;

    lspmac_next_state();

    if( ls_pmac_state != old_state ) {
        //      lslogging_log_message( "lspmac_worker: state = %d",
        //      ls_pmac_state);
        old_state = ls_pmac_state;
    }

    if( pmacfd.fd == -1 ) {
        if( disconnected_notify == 0 )
            lslogging_log_message( "lspmac_worker: PMAC not
            connected");
        disconnected_notify = 1;
        //
        // At this point we assume we became disconnected due to something like a
        // hard boot of the MD2 PMAC
        // and hence the entire system needs reinitialization.
        //
        // It's possible to put in a test here (perhaps using I65) to see if we
        // in fact suffered a reset
        // and need to clear the queue, reinitialize, etc. Or if it was just a
        // networking glitch and do not
        // need to clear the queue and should instead just charge ahead.
        //
        lspmac_reset_queue();
        sleep( 10);
        //
        // This just puts us into a holding pattern until the pmac becomes
        // connected again
        //
        continue;
    }
    disconnected_notify = 0;

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

```

7.6.4.87 void lspmac_zoom_lut_setup ()

Set up lookup table for zoom.

Definition at line 3359 of file lspmac.c.

```

{
    int i;
    lsredis_obj_t *p;

    pthread_mutex_lock( &zoom->mutex);

    zoom->nlut = 10;
    zoom->lut = calloc( 2 * zoom->nlut, sizeof( double));
    if( zoom->lut == NULL ) {
        lslogging_log_message( "lspmac_zoom_lut_setup: out of
        memory");
        exit( -1);
    }

    for( i=0; i < zoom->nlut; i++) {
        p = lsredis_get_obj( "cam.zoom.%d.MotorPosition", i+1);
        if( p==NULL || strlen( lsredis_getstr(p)) == 0 ) {
            free( zoom->lut);
            zoom->lut = NULL;
            zoom->nlut = 0;
            pthread_mutex_unlock( &zoom->mutex);
            lslogging_log_message( "lspmac_zoom_lut_setup:
            cannot find MotorPosition element for cam.zoom level %d", i+1);
            return;
        }
        zoom->lut[2*i] = i+1;
        zoom->lut[2*i+1] = lsredis_getd( p);
    }
    pthread_mutex_unlock( &zoom->mutex);
}

```


7.6.5 Variable Documentation

7.6.5.1 Ispmac_motor_t* alignx

Alignment stage X.

Definition at line 90 of file Ispmac.c.

7.6.5.2 Ispmac_motor_t* aligny

Alignment stage Y.

Definition at line 91 of file Ispmac.c.

7.6.5.3 Ispmac_motor_t* alignz

Alignment stage X.

Definition at line 92 of file Ispmac.c.

7.6.5.4 Ispmac_motor_t* anal

Polaroid analyzer motor.

Definition at line 93 of file Ispmac.c.

7.6.5.5 Ispmac_motor_t* apery

Aperture Y.

Definition at line 95 of file Ispmac.c.

7.6.5.6 Ispmac_motor_t* aperz

Aperture Z.

Definition at line 96 of file Ispmac.c.

7.6.5.7 Ispmac_bi_t* arm_parked

(whose arm? parked where?)

Definition at line 133 of file Ispmac.c.

7.6.5.8 Ispmac_motor_t* blight

Back Light DAC.

Definition at line 107 of file Ispmac.c.

7.6.5.9 Ispmac_bi_t* blight_down

Backlight is down.

Definition at line 123 of file Ispmac.c.

7.6.5.10 `lspmac_motor_t* blight_f`

Back light scale factor.

Definition at line 116 of file `lspmac.c`.

7.6.5.11 `lspmac_motor_t* blight_ud`

Back light Up/Down actuator.

Definition at line 111 of file `lspmac.c`.

7.6.5.12 `lspmac_bi_t* blight_up`

Backlight is up.

Definition at line 124 of file `lspmac.c`.

7.6.5.13 `lspmac_motor_t* capy`

Capillary Y.

Definition at line 97 of file `lspmac.c`.

7.6.5.14 `lspmac_motor_t* capz`

Capillary Z.

Definition at line 98 of file `lspmac.c`.

7.6.5.15 `lspmac_motor_t* cenx`

Centering Table X.

Definition at line 100 of file `lspmac.c`.

7.6.5.16 `lspmac_motor_t* ceny`

Centering Table Y.

Definition at line 101 of file `lspmac.c`.

7.6.5.17 `pmac_cmd_t cr_cmd` `[static]`

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

Definition at line 185 of file `lspmac.c`.

7.6.5.18 `lspmac_motor_t* cryo`

Move the cryostream towards or away from the crystal.

Definition at line 112 of file `lspmac.c`.

7.6.5.19 lspmac_bi_t* cryo_back

cryo is in the back position

Definition at line 125 of file lspmac.c.

7.6.5.20 lspmac_bi_t* cryo_switch

that little toggle switch for the cryo

Definition at line 122 of file lspmac.c.

7.6.5.21 unsigned char dbmem[64 * 1024] [static]

double buffered memory

Definition at line 174 of file lspmac.c.

7.6.5.22 int dbmemIn = 0 [static]

next location

Definition at line 175 of file lspmac.c.

7.6.5.23 lspmac_motor_t* dryer

blow air on the scintillator to dry it off

Definition at line 113 of file lspmac.c.

7.6.5.24 lspmac_bi_t* etel_init_ok

ETEL initialized OK.

Definition at line 130 of file lspmac.c.

7.6.5.25 lspmac_bi_t* etel_on

ETEL is on.

Definition at line 129 of file lspmac.c.

7.6.5.26 lspmac_bi_t* etel_ready

ETEL is ready.

Definition at line 128 of file lspmac.c.

7.6.5.27 unsigned int ethCmdOff = 0 [static]

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

Definition at line 188 of file lspmac.c.

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

points to next empty PMAC command queue position

Definition at line 187 of file `lspmac.c`.

7.6.5.29 `pmac_cmd_queue_t ethCmdQueue[PMAC_CMD_QUEUE_LENGTH]` `[static]`

PMAC command queue.

Definition at line 186 of file `lspmac.c`.

7.6.5.30 `unsigned int ethCmdReply = 0` `[static]`

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

Definition at line 189 of file `lspmac.c`.

7.6.5.31 `lspmac_motor_t* flight`

Front Light DAC.

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

7.6.5.32 `lspmac_motor_t* flight.f`

Front light scale factor.

Definition at line 117 of file `lspmac.c`.

7.6.5.33 `lspmac_motor_t* flight.oo`

Turn front light on/off.

Definition at line 115 of file `lspmac.c`.

7.6.5.34 `lspmac_motor_t* fluo`

Move the fluorescence detector in/out.

Definition at line 114 of file `lspmac.c`.

7.6.5.35 `lspmac_bi_t* fluor.back`

fluor is in the back position

Definition at line 126 of file `lspmac.c`.

7.6.5.36 `lspmac_motor_t* fscint`

Scintillator Piezo DAC.

Definition at line 108 of file `lspmac.c`.

7.6.5.37 lspmac_motor_t* fshut

Fast shutter.

Definition at line 105 of file lspmac.c.

7.6.5.38 pmac_cmd_t gb_cmd [static]

Definition at line 185 of file lspmac.c.

7.6.5.39 int getivars = 0 [static]

flag set at initialization to send i vars to db

Definition at line 81 of file lspmac.c.

7.6.5.40 int getmvars = 0 [static]

flag set at initialization to send m vars to db

Definition at line 82 of file lspmac.c.

7.6.5.41 lspmac_bi_t* hp_air

High pressure air OK.

Definition at line 121 of file lspmac.c.

7.6.5.42 lspmac_motor_t* kappa

Kappa.

Definition at line 102 of file lspmac.c.

7.6.5.43 lspmac_bi_t* lp_air

Low pressure air OK.

Definition at line 120 of file lspmac.c.

7.6.5.44 int ls_pmac_state = LS_PMAC_STATE_DETACHED [static]

Current state of the PMAC communications state machine.

Definition at line 51 of file lspmac.c.

7.6.5.45 lspmac_ascii_buffers_t lspmac_ascii_buffers [static]

Definition at line 354 of file lspmac.c.

7.6.5.46 pthread_mutex_t lspmac_ascii_buffers_mutex

Definition at line 355 of file lspmac.c.

7.6.5.47 `int lspmac_ascii_busy = 0` `[static]`

flag for condition to wait for

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

7.6.5.48 `pthread_mutex_t lspmac_ascii_mutex` `[static]`

Keep too many processes from sending commands at once.

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

7.6.5.49 `lspmac_bi_t lspmac_bis[32]`

array of binary inputs

Definition at line 84 of file `lspmac.c`.

7.6.5.50 `uint32_t lspmac_dpascii_off = 0` `[static]`

Definition at line 365 of file `lspmac.c`.

7.6.5.51 `uint32_t lspmac_dpascii_on = 0` `[static]`

Definition at line 364 of file `lspmac.c`.

7.6.5.52 `lspmac_dpascii_queue_t lspmac_dpascii_queue[LSPMAC_DPASCII_QUEUE_LENGTH]` `[static]`

Definition at line 363 of file `lspmac.c`.

7.6.5.53 `lspmac_motor_t lspmac_motors[48]`

All our motors.

Definition at line 87 of file `lspmac.c`.

7.6.5.54 `pthread_cond_t lspmac_moving_cond`

Wait for motor(s) to finish moving condition.

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

7.6.5.55 `int lspmac_moving_flags`

Flag used to implement motor moving condition.

Definition at line 65 of file `lspmac.c`.

7.6.5.56 `pthread_mutex_t lspmac_moving_mutex`

Coordinate moving motors between threads.

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

7.6.5.57 `int lspmac_nbis = 0`

number of active binary inputs

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

7.6.5.58 `int lspmac_nmotors = 0`

The number of motors we manage.

Definition at line 88 of file `lspmac.c`.

7.6.5.59 `pthread_cond_t lspmac_shutter_cond`

Allows waiting for the shutter status to change.

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

7.6.5.60 `int lspmac_shutter_has_opened`

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

Definition at line 60 of file `lspmac.c`.

7.6.5.61 `pthread_mutex_t lspmac_shutter_mutex`

Coordinates threads reading shutter status.

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

7.6.5.62 `int lspmac_shutter_state`

State of the shutter, used to detect changes.

Definition at line 59 of file `lspmac.c`.

7.6.5.63 `struct timespec lspmac_status_last_time` `[static]`

Time the status was read.

Definition at line 74 of file `lspmac.c`.

7.6.5.64 `struct timespec lspmac_status_time` `[static]`

Time the status was read.

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

7.6.5.65 `md2_status_t md2_status` `[static]`

Buffer for MD2 Status.

Definition at line 340 of file `lspmac.c`.

7.6.5.66 pthread_mutex_t md2_status_mutex

Synchronize reading/writing status buffer.

Definition at line 341 of file lspmac.c.

7.6.5.67 lspmac_bi_t* minikappa_ok

Minikappa is OK (whatever that means)

Definition at line 131 of file lspmac.c.

7.6.5.68 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 181 of file lspmac.c.

7.6.5.69 lspmac_motor_t* omega

MD2 omega axis (the air bearing)

Definition at line 89 of file lspmac.c.

7.6.5.70 int omega_zero_search = 0 [static]

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

Definition at line 70 of file lspmac.c.

7.6.5.71 struct timespec omega_zero_time

Time we believe that omega crossed zero.

Definition at line 72 of file lspmac.c.

7.6.5.72 double omega_zero_velocity = 0 [static]

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

Definition at line 71 of file lspmac.c.

7.6.5.73 lspmac_motor_t* phi

Phi (not data collection axis)

Definition at line 103 of file lspmac.c.

7.6.5.74 char* pmac_error_strs[] [static]

Initial value:

```
= {
    "ERR000: Unknown error",
    "ERR001: Command not allowed during program execution",
    "ERR002: Password error",
    "ERR003: Data error or unrecognized command",
    "ERR004: Illegal character",
```



```

"ERR005: Command not allowed unless buffer is open",
"ERR006: No room in buffer for command",
"ERR007: Buffer already in use",
"ERR008: MACRO auziliary communication error",
"ERR009: Program structure error (e.g. ENDIF without IF)",
"ERR010: Both overtravel limits set for a motor in the C.S.",
"ERR011: Previous move not completed",
"ERR012: A motor in the coordinate system is open-loop",
"ERR013: A motor in the coordinate system is not activated",
"ERR014: No motors in the coordinate system",
"ERR015: Not pointer to valid program buffer",
"ERR016: Running improperly structure program (e.g. missing ENDWHILE)",
"ERR017: Trying to resume after H or Q with motors out of stopped position",
"ERR018: Attempt to perform phase reference during move, move during phase
reference, or enabling with phase clock error",
"ERR019: Illegal position-chage command while moves stored in CCBUFFER",
"ERR020: FSAVE issued on Turbo PMAC with incompatible flash memory",
"ERR021: FSAVE issued while clearing old flash memory sector",
"ERR022: FREAD attempted but the flash memory is bad"
}

```

Decode the errors perhaps returned by the PMAC.

Definition at line 192 of file lspmac.c.

7.6.5.75 pthread_cond_t pmac_queue_cond

wait for a command to be sent to PMAC before continuing

Definition at line 78 of file lspmac.c.

7.6.5.76 pthread_mutex_t pmac_queue_mutex

manage access to the pmac command queue

Definition at line 77 of file lspmac.c.

7.6.5.77 pthread_t pmac_thread [static]

our thread to manage access and communication to the pmac

Definition at line 76 of file lspmac.c.

7.6.5.78 struct pollfd pmacfd [static]

our poll structure

Definition at line 79 of file lspmac.c.

7.6.5.79 pmac_cmd_t rr_cmd [static]

Definition at line 185 of file lspmac.c.

7.6.5.80 lspmac_bi_t* sample_detected

smart magnet detected sample

Definition at line 127 of file lspmac.c.

7.6.5.81 lspmac_motor_t* scint

Scintillator Z.

Definition at line 99 of file lspmac.c.

7.6.5.82 `lspmac_bi_t* shutter_open`

shutter is open (note in pmc says this is a slow input)

Definition at line 134 of file lspmac.c.

7.6.5.83 `lspmac_bi_t* smart_mag_err`

smart magnet error (coil broken perhaps)

Definition at line 135 of file lspmac.c.

7.6.5.84 `lspmac_bi_t* smart_mag_off`

smart magnet is off

Definition at line 136 of file lspmac.c.

7.6.5.85 `lspmac_bi_t* smart_mag_on`

smart magnet is on

Definition at line 132 of file lspmac.c.

7.6.5.86 `lspmac_motor_t* smart_mag_oo`

Smart Magnet on/off.

Definition at line 110 of file lspmac.c.

7.6.5.87 `lspmac_motor_t* zoom`

Optical zoom.

Definition at line 94 of file lspmac.c.

7.7 `lsredis.c` File Reference

Support redis hash synchronization.

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

Functions

- void `lsredis_debugCB` (`redisAsyncContext *ac`, void `*reply`, void `*privdata`)
Log the reply.
- void `__lsredis_set_value` (`lsredis_obj_t *p`, char `*v`)
set_value and setstr helper function p->mutex must be locked before calling
- void `lsredis_set_value` (`lsredis_obj_t *p`, char `*fmt`,...)
Set the value of a redis object and make it valid.
- int `lsredis_cmpstr` (`lsredis_obj_t *p`, char `*s`)

- int [lsredis_cmpnstr](#) ([lsredis_obj_t](#) *p, char *s, int n)
- int [lsredis_regexec](#) (const regex_t *preg, [lsredis_obj_t](#) *p, size_t nmatch, regmatch_t *pmatch, int eflags)
- char * [lsredis_getstr](#) ([lsredis_obj_t](#) *p)
 - return a copy of the key's string value*
- void [lsredis_setstr](#) ([lsredis_obj_t](#) *p, char *fmt,...)
 - Set the value and update redis.*
- double [lsredis_getd](#) ([lsredis_obj_t](#) *p)
- long int [lsredis_getl](#) ([lsredis_obj_t](#) *p)
- char ** [lsredis_get_string_array](#) ([lsredis_obj_t](#) *p)
- int [lsredis_getb](#) ([lsredis_obj_t](#) *p)
- char [lsredis_getc](#) ([lsredis_obj_t](#) *p)
- void [lsredis_hgetCB](#) (redisAsyncContext *ac, void *reply, void *privdata)
- [lsredis_obj_t](#) * [_lsredis_get_obj](#) (char *key)
 - Maybe add a new object Used internally for this module Must be called with lsredis_mutex locked.*
- [lsredis_obj_t](#) * [lsredis_get_obj](#) (char *fmt,...)
- void [redisDisconnectCB](#) (const redisAsyncContext *ac, int status)
 - call back in case a redis server becomes disconnected TODO: reconnect*
- void [lsredis_addRead](#) (void *data)
 - hook to mange read events*
- void [lsredis_delRead](#) (void *data)
 - hook to manage "don't need to read" events*
- void [lsredis_addWrite](#) (void *data)
 - hook to manage write events*
- void [lsredis_delWrite](#) (void *data)
 - hook to manage "don't need to write anymore" events*
- void [lsredis_cleanup](#) (void *data)
 - hook to clean up TODO: figure out what we are supposed to do here and do it*
- void [lsredis_subCB](#) (redisAsyncContext *ac, void *reply, void *privdata)
 - Use the publication to request the new value.*
- void [lsredis_maybe_add_key](#) (char *k)
- void [lsredis_keysCB](#) (redisAsyncContext *ac, void *reply, void *privdata)
 - Sift through the keys to find ones we like.*
- int [lsredis_find_preset](#) (char *base, char *preset_name, double *dval)
- void [lsredis_init](#) (char *pub, char *re, char *head)
 - Initialize this module, that is, set up the connections.*
- void [lsredis_fd_service](#) (struct pollfd *evt)
 - service the socket requests*
- void [lsredis_sig_service](#) (struct pollfd *evt)
- void * [lsredis_worker](#) (void *dummy)
 - subscribe to changes and service sockets*
- void [lsredis_run](#) ()

Variables

- static pthread_t [lsredis_thread](#)
- static pthread_mutex_t [lsredis_mutex](#) = PTHREAD_RECURSIVE_MUTEX_INITIALIZER_NP
- static pthread_cond_t [lsredis_cond](#)
- static int [lsredis_running](#) = 0
- static [lsredis_obj_t](#) * [lsredis_objs](#) = NULL
- static struct hsearch_data [lsredis_htab](#)
- static redisAsyncContext * [subac](#)
- static redisAsyncContext * [roac](#)

- static redisAsyncContext * [wrac](#)
- static char * [lsredis_publisher](#) = NULL
- static regex_t [lsredis_key_select_regex](#)
- static char * [lsredis_head](#) = NULL
- static struct pollfd [subfd](#)
- static struct pollfd [rofd](#)
- static struct pollfd [wrfd](#)

7.7.1 Detailed Description

Support redis hash synchronization.

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

Redis support for redis in pgpmac.

Values in redis are assumed to be hashes with at list one field "VALUE". At startup the initialization routine is passed a regular expression to select which keys we'd like to duplicate locally as a `lsredis_obj_t`. It is assumed that the following construct in redis is used to change a value:

```
MULTI
HSET key VALUE value
PUBLISH publisher key
EXEC
```

Where "publisher" is a unique name in the following format:

```
MD2-*
or    UI-*
or    REDIS_KV_CONNECTOR
```

(this last value is used to support the now depreciated `px.kvs` table in the LS-CAT postgresql server). We assume that all publisher that we are listening to ONLY publish key names that have changed.

When someone else changes a value we invalidate our internal copy and issue a "HGET key VALUE" command. Other threads that request the value of our `lsredis_obj_t` will pause until the new value has been received and processed.

When a value changes locally this module changes it in redis as shown above. At this point we refuse other publishers attempt to change the value until we've seen all of our PUBLISH messages. That is, we ignore changes that in redis happened before our change.

You'll need an `lsredis_obj_t` to do anything with redis in the pgpmac project:

```
lsredis_obj_t *lsredis_get_obj( char *fmt, ...) where fmt is a printf style formatting string
During initialization a "head" string is passed
For example, "omega.position" might refer to t
```

To set a redis value use

```
void lsredis_setstr( lsredis_obj_t *p, char *fmt, ...) where fmt is a printf style formatting
```

When a new value is seen we immediately parse it and make it available through the following functions:

char	<code>*lsredis_getstr(lsredis_obj_t *p)</code>	Returns a copy of the VALUE field. Use
double	<code>lsredis_getd(lsredis_obj_t *p)</code>	Returns a double. If the value was not
long int	<code>lsredis_getl(lsredis_obj_t *p)</code>	Returns a long int. If the value was not
char	<code>**lsredis_get_string_array(lsredis_obj_t *p)</code>	Returns an array of string pointers. V or NULL if the value could not be parsed
int	<code>lsredis_getb(lsredis_obj_t *p)</code>	Returns 1, 0, or -1 based on the first c
char	<code>lsredis_getc(lsredis_obj_t *p)</code>	Returns the first character of VALUE

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

7.7.2 Function Documentation

7.7.2.1 `lsredis_obj_t* lsredis_get_obj(char * key)`

Maybe add a new object Used internally for this module Must be called with `lsredis_mutex` locked.

Definition at line 438 of file `lsredis.c`.

```

{
    lsredis_obj_t *p;
    regmatch_t pmatch[2];
    int err;
    ENTRY htab_input, *htab_output;

    // Dispense with obviously bad keys straight away
    // unless p->valid == 0 in which case we call HGET first
    //
    // TODO: review logic: is there ever a time when valid is zero for a
    // preexisting p and HGET has not been called?
    // If not then we should just return p without checking for validity.
    //
    if( key == NULL || *key == 0 || strchr( key, ' ' ) != NULL ) {
        lslogging_log_message( "_lsredis_get_obj: bad key '%s'"
            , key == NULL ? "<NULL>" : key );
        return NULL;
    }

    // If the key is already there then just return it
    //

    htab_input.key = key;
    htab_input.data = NULL;
    errno = 0;
    err = hsearch_r( htab_input, FIND, &htab_output, &lsredis_htab );

    if( err == 0 )
        p = NULL;
    else
        p = htab_output->data;

    if( p != NULL ) {
        return p;
    } else {
        // make a new one.
        p = calloc( 1, sizeof( lsredis_obj_t ) );
        if( p == NULL ) {
            lslogging_log_message( "_lsredis_get_obj: Out of
                memory" );
            exit( -1 );
        }

        err = regexec( &lsredis_key_select_regex, key, 2,
            pmatch, 0 );
        if( err == 0 && pmatch[1].rm_so != -1 ) {

```

```

    p->events_name = strdup( key+pmatch[1].rm_so, pmatch[1].rm_eo
    - pmatch[1].rm_so);
} else {
    p->events_name = strdup( key);
}
if( p->events_name == NULL) {
    lslogging_log_message( "_lsredis_get_obj: Out of
    memory (events_name)");
    exit( -1);
}

pthread_mutex_init( &p->mutex, NULL);
pthread_cond_init( &p->cond, NULL);
p->value = NULL;
p->valid = 0;
lsevents_send_event( "%s Invalid", p->events_name
);
p->wait_for_me = 0;
p->key = strdup( key);
p->hits = 0;

htab_input.key = p->key;
htab_input.data = p;

errno = 0;
err = hsearch_r( htab_input, ENTER, &htab_output, &lsredis_htab
);
if( err == 0) {
    lslogging_log_message( "_lsredis_get_obj: hsearch
    error on enter.  errno=%d", errno);
}

//
// Shouldn't need the linked list unless we need to rebuild the hash table
// when, for example, we run out of room.
// TODO: resize hash table when needed.
//
p->next = lsredis_objs;
lsredis_objs = p;
}
//
// We arrive here with the valid flag lowered.  Go ahead and request the
// latest value.
//
redisAsyncCommand( roac, lsredis_hgetCB, p, "HGET %s VALUE"
, key);

return p;
}

```

7.7.2.2 void lsredis_set_value(lsredis_obj_t *p, char *v)

set_value and setstr helper function p->mutex must be locked before calling

Definition at line 146 of file lsredis.c.

```

{
if( strlen(v) >= (unsigned int) p->value_length) {
    if( p->value != NULL)
        free( p->value);
    p->value_length = strlen(v) + 256;
    p->value = calloc( p->value_length, sizeof( char));
    if( p->value == NULL) {
        lslogging_log_message( "_lsredis_set_value: out of
        memory");
        exit( -1);
    }
}
strncpy( p->value, v, p->value_length - 1);
p->value[p->value_length-1] = 0;
p->dvalue = strtod( p->value, NULL);
p->lvalue = p->dvalue;

if( p->avalue != NULL) {
    int i;
    for( i=0; (p->avalue)[i] != NULL; i++)
        free( (p->avalue)[i]);
    free( p->avalue);
    p->avalue = NULL;
}
}

```

```

p->avalue = lspg_array2ptrs( p->value);

switch( *(p->value)) {
    case 'T':
    case 't':
    case 'Y':
    case 'y':
    case 'l':
        p->bvalue = 1;
        break;

    case 'F':
    case 'f':
    case 'N':
    case 'n':
    case '0':
        p->bvalue = 0;
        break;

    default:
        p->bvalue = -1;           // nil is -1 here in our world
}

p->cvalue = *(p->value);

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

```

7.7.2.3 void Isredis_addRead (void * data)

hook to mange read events

Definition at line 568 of file Isredis.c.

```

{
    struct pollfd *pfd;
    pfd = (struct pollfd *)data;

    if( (pfd->events & POLLIN) == 0) {
        pfd->events |= POLLIN;
        pthread_kill( lsredis_thread, SIGUSR1);
    }
}

```

7.7.2.4 void Isredis_addWrite (void * data)

hook to manage write events

Definition at line 592 of file Isredis.c.

```

{
    struct pollfd *pfd;
    pfd = (struct pollfd *)data;

    if( (pfd->events & POLLOUT) == 0) {
        pfd->events |= POLLOUT;
        pthread_kill( lsredis_thread, SIGUSR1);
    }
}

```

7.7.2.5 void Isredis_cleanup (void * data)

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

Definition at line 617 of file Isredis.c.

```

{
    struct pollfd *pfd;
    pfd = (struct pollfd *)data;

    pfd->fd = -1;

    if( (pfd->events & (POLLOUT | POLLIN)) != 0 ) {
        pfd->events &= ~(POLLOUT | POLLIN);
        pthread_kill( lsredis_thread, SIGUSR1);
    }
}

```

7.7.2.6 int lsredis_cmpnstr (lsredis_obj_t *p, char *s, int n)

Definition at line 236 of file lsredis.c.

```

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

    rtn = strncmp( p->value, s, n);
    pthread_mutex_unlock( &p->mutex);
    return rtn;
}

```

7.7.2.7 int lsredis_cmpstr (lsredis_obj_t *p, char *s)

Definition at line 225 of file lsredis.c.

```

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

    rtn = strcmp( p->value, s);
    pthread_mutex_unlock( &p->mutex);
    return rtn;
}

```

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

Log the reply.

Definition at line 96 of file lsredis.c.

```

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

    r = (redisReply *)reply;

    if( r == NULL ) {
        lslogging_log_message( "Null reply. Odd");
        return;
    }

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

    case REDIS_REPLY_ERROR:
        lslogging_log_message( "%sERROR: %s", indentlevel*4,
            "", r->str);
        break;
    }
}

```



```

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

case REDIS_REPLY_NIL:
    lslogging_log_message( "%s(nil)", indentlevel*4, "");
    break;

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

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

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

```

7.7.2.9 void lsredis_delRead (void * *data*)

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

Definition at line 580 of file Isredis.c.

```

{
    struct pollfd *pfd;
    pfd = (struct pollfd *)data;

    if( (pfd->events & POLLIN) != 0) {
        pfd->events &= ~POLLIN;
        pthread_kill( lsredis_thread, SIGUSR1);
    }
}

```

7.7.2.10 void lsredis_delWrite (void * *data*)

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

Definition at line 604 of file Isredis.c.

```

{
    struct pollfd *pfd;
    pfd = (struct pollfd *)data;

    if( (pfd->events & POLLOUT) != 0) {
        pfd->events &= ~POLLOUT;
        pthread_kill( lsredis_thread, SIGUSR1);
    }
}

```

7.7.2.11 void lsredis_fd_service (struct pollfd * *evt*)

service the socket requests

Definition at line 889 of file Isredis.c.

```

{
    pthread_mutex_lock( &lsredis_mutex);
    if( evt->fd == subac->c.fd) {

```

```

    if( evt->revents & POLLIN)
        redisAsyncHandleRead( subac);
    if( evt->revents & POLLOUT)
        redisAsyncHandleWrite( subac);
}
if( evt->fd == roac->c.fd) {
    if( evt->revents & POLLIN)
        redisAsyncHandleRead( roac);
    if( evt->revents & POLLOUT)
        redisAsyncHandleWrite( roac);
}
if( evt->fd == wrac->c.fd) {
    if( evt->revents & POLLIN)
        redisAsyncHandleRead( wrac);
    if( evt->revents & POLLOUT)
        redisAsyncHandleWrite( wrac);
}
pthread_mutex_unlock( &lsredis_mutex);
}

```

7.7.2.12 int lsredis_find_preset(char * base, char * preset_name, double * dval)

Definition at line 756 of file lsredis.c.

```

char s[512];
int i;
int err;
ENTRY htab_input, *htab_output;
lsredis_obj_t *p;

i = 0;
for( i=0; i<1024; i++) {
    snprintf( s, sizeof( s)-1, "%s.%s.presets.%d.name", lsredis_head
, base, i);
    s[sizeof(s)-1] = 0;
    htab_input.key = s;
    htab_input.data = NULL;
    err = hsearch_r( htab_input, FIND, &htab_output, &lsredis_htab
);
    if( err == 0) {
        // We've run out of names to look for: done
        lslogging_log_message( "lsredis_find_preset: no
        preset for motor %s named '%s'", base, preset_name);
        *dval = 0.0;
        return 0;
    }

    // Check if we have a match
    p = htab_output->data;
    if( lsredis_cmpstr( p, preset_name)==0) {
        // got a match, now look for the position
        snprintf( s, sizeof( s)-1, "%s.%s.presets.%d.position", lsredis_head
, base, i);
        s[sizeof(s)-1] = 0;
        htab_input.key = s;
        htab_input.data = NULL;
        err = hsearch_r( htab_input, FIND, &htab_output, &lsredis_htab
);
        if( err == 0) {
            // Name but not position? odd.
            lslogging_log_message( "lsredis_find_preset:
            Error, motor %s preset '%s' has no position defined", base, preset_name);
            *dval = 0.0;
            return 0;
        }
        p = htab_output->data;
        *dval = lsredis_getd( p);
        return 1;
    }
}
// How'd we get here?
// did someone really define that many presets? And then looked for one
// that's not there?
*dval = 0;
return 0;
}

```

7.7.2.13 `lsredis_obj_t* lsredis_get_obj(char *fmt, ...)`

Definition at line 524 of file `lsredis.c`.

```

{
    lsredis_obj_t *rtn;
    va_list arg_ptr;
    char k[512];
    char *kp;
    int nkp;

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

    nkp = strlen(k) + strlen( lsredis_head) + 16;           // 16
                is overkill. I know. Get over it.
    kp = calloc( nkp, sizeof( char));
    if( kp == NULL) {
        lslogging_log_message( "lsredis_get_obj: Out of memory
                                ");
        exit( -1);
    }

    snprintf( kp, nkp-1, "%s.%s", lsredis_head, k);
    kp[nkp-1] = 0;

    pthread_mutex_lock( &lsredis_mutex);
    while( lsredis_running == 0)
        pthread_cond_wait( &lsredis_cond, &lsredis_mutex);

    rtn = _lsredis_get_obj( kp);
    pthread_mutex_unlock( &lsredis_mutex);

    free( kp);
    return rtn;
}

```

7.7.2.14 `char** lsredis_get_string_array(lsredis_obj_t *p)`

Definition at line 365 of file `lsredis.c`.

```

{
    char **rtn;

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

    rtn = p->avalue;
    pthread_mutex_unlock( &p->mutex);

    return rtn;
}

```

7.7.2.15 `int lsredis_getb(lsredis_obj_t *p)`

Definition at line 378 of file `lsredis.c`.

```

{
    int rtn;

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

    rtn = p->bvalue;
    pthread_mutex_unlock( &p->mutex);

    return rtn;
}

```

7.7.2.16 char lsredis_getc (lsredis_obj_t * p)

Definition at line 391 of file lsredis.c.

```

{
    int rtn;

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

    rtn = p->cvalue;
    pthread_mutex_unlock( &p->mutex);

    return rtn;
}

```

7.7.2.17 double lsredis_getd (lsredis_obj_t * p)

Definition at line 339 of file lsredis.c.

```

{
    double rtn;

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

    rtn = p->dvalue;
    pthread_mutex_unlock( &p->mutex);

    return rtn;
}

```

7.7.2.18 long int lsredis_getl (lsredis_obj_t * p)

Definition at line 352 of file lsredis.c.

```

{
    long int rtn;

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

    rtn = p->lvalue;
    pthread_mutex_unlock( &p->mutex);

    return rtn;
}

```

7.7.2.19 char* lsredis_getstr (lsredis_obj_t * p)

return a copy of the key's string value

Definition at line 263 of file lsredis.c.

```

{
    char *rtn;

    //
    // Have to use strdup since we cannot guarantee that p->value won't be freed
    // while the caller is still using it
    //
    pthread_mutex_lock( &p->mutex);
    while( p->valid == 0)
        pthread_cond_wait( &p->cond, &p->mutex);

    rtn = strdup(p->value);
    pthread_mutex_unlock( &p->mutex);
    return rtn;
}

```

7.7.2.20 void Isredis_hgetCB (redisAsyncContext * *ac*, void * *reply*, void * *privdata*)

Definition at line 404 of file Isredis.c.

```

{
    redisReply *r;
    lsredis_obj_t *p;

    r = reply;
    p = privdata;

    // lslogging_log_message( "hgetCB: %s %s", p == NULL ? "<NULL>" : p->key,
        r->type == REDIS_REPLY_STRING ? r->str : "Non-string value. Why?");

    //
    // Apparently this item does not exist
    // Just set it to an empty string so at least other apps will have the same
        behaviour as us
    // TODO: figure out a better way to deal with missing key/values
    //
    if( p != NULL && r->type == REDIS_REPLY_NIL) {
        lsredis_setstr( p, "");
        return;
    }

    if( p != NULL && r->type == REDIS_REPLY_STRING && r->str != NULL) {
        pthread_mutex_lock( &p->mutex);

        _lsredis_set_value( p, r->str);

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

```

7.7.2.21 void Isredis_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 810 of file Isredis.c.

```

{
    int err;
    int nerrmsg;
    char *errmsg;

    //
    // set up hash map to store redis objects
    //
    err = hcreate_r( 8192, &lsredis_hstab);
    if( err == 0) {
        lslogging_log_message( "lsredis_init: Cannot create
            hash table. Really bad things are going to happen. hcreate_r returned %d", err);
    }

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

    pthread_cond_init( &lsredis_cond, NULL);

    subac = redisAsyncConnect("127.0.0.1", 6379);
    if( subac->err) {
        lslogging_log_message( "Error: %s", subac->errstr
            );
    }

    subfd.fd = subac->c.fd;
    subfd.events = 0;
    subac->ev.data = &subfd;
}

```

```

subac->ev.addRead = lsredis_addRead;
subac->ev.delRead = lsredis_delRead;
subac->ev.addWrite = lsredis_addWrite;
subac->ev.delWrite = lsredis_delWrite;
subac->ev.cleanup = lsredis_cleanup;

roac = redisAsyncConnect("127.0.0.1", 6379);
if( roac->err) {
    lslogging_log_message( "Error: %s", roac->errstr);
}

rofd.fd = roac->c.fd;
rofd.events = 0;
roac->ev.data = &rofd;
roac->ev.addRead = lsredis_addRead;
roac->ev.delRead = lsredis_delRead;
roac->ev.addWrite = lsredis_addWrite;
roac->ev.delWrite = lsredis_delWrite;
roac->ev.cleanup = lsredis_cleanup;

//wrac = redisAsyncConnect("10.1.0.3", 6379);
wrac = redisAsyncConnect("127.0.0.1", 6379);
if( wrac->err) {
    lslogging_log_message( "Error: %s", wrac->errstr);
}

wrfd.fd = wrac->c.fd;
wrfd.events = 0;
wrac->ev.data = &wrfd;
wrac->ev.addRead = lsredis_addRead;
wrac->ev.delRead = lsredis_delRead;
wrac->ev.addWrite = lsredis_addWrite;
wrac->ev.delWrite = lsredis_delWrite;
wrac->ev.cleanup = lsredis_cleanup;

err = regcomp( &lsredis_key_select_regex, re,
    REG_EXTENDED);
if( err != 0) {
    nerrmsg = regerror( err, &lsredis_key_select_regex,
        NULL, 0);
    if( nerrmsg > 0) {
        errmsg = calloc( nerrmsg, sizeof( char));
        nerrmsg = regerror( err, &lsredis_key_select_regex
            , errmsg, nerrmsg);
        lslogging_log_message( "lsredis_select: %s", errmsg);
        free( errmsg);
    }
}
}
}

```

7.7.2.22 void lsredis_keysCB (redisAsyncContext * ac, void * reply, void * privdata)

Sift through the keys to find ones we like.

Add them to our list of followed objects

Definition at line 735 of file lsredis.c.

```

redisReply *r;
int i;

r = reply;
if( r->type != REDIS_REPLY_ARRAY) {
    lslogging_log_message( "lsredis_keysCB: expected
        array...");
    lsredis_debugCB( ac, reply, privdata);
    return;
}

for( i=0; i< (int)r->elements; i++) {
    if( r->element[i]->type != REDIS_REPLY_STRING) {
        lslogging_log_message( "lsredis_keysCB: expected
            string...");
        lsredis_debugCB( ac, r->element[i], privdata);
    } else {
        lsredis_maybe_add_key( r->element[i]->str);
    }
}
}

```

7.7.2.23 void Isredis_maybe_add_key (char * *k*)

Definition at line 727 of file Isredis.c.

```

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

```

7.7.2.24 int Isredis_regexec (const regex_t * *preg*, Isredis_obj_t * *p*, size_t *nmatch*, regmatch_t * *pmatch*, int *eflags*)

Definition at line 247 of file Isredis.c.

```

{
    int rtn;

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

    rtn = regexec( preg, p->value, nmatch, pmatch, eflags);

    pthread_mutex_unlock( &p->mutex);

    return rtn;
}

```

7.7.2.25 void Isredis_run ()

Definition at line 1014 of file Isredis.c.

```

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

```

7.7.2.26 void Isredis_set_value (Isredis_obj_t * *p*, char * *fmt*, ...)

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

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

Definition at line 207 of file Isredis.c.

```

{
    va_list arg_ptr;
    char v[512];

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

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

    pthread_mutex_lock( &p->mutex);

    _lsredis_set_value( p, v);

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

```

7.7.2.27 void lsredis_setstr (lsredis_obj_t * p, char * fmt, ...)

Set the value and update redis.

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

redisAsyncCommandArgv used instead of redisAsyncCommand 'cause it's easier (and possible) to deal with strings that would otherwise cause hiredis to emit a bad command, like those containing spaces. < up the count of times we need to see ourselves published before we start listening to others again

< Unlock to prevent deadlock in case the service routine needs to set our value

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

Definition at line 288 of file lsredis.c.

```

{
    va_list arg_ptr;
    char v[512];
    char *argv[4];

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

    pthread_mutex_lock( &p->mutex);

    //
    // Don't send an update if a good value has not changed
    //
    if( p->valid && strcmp( v, p->value) == 0) {
        // nothing to do
        pthread_mutex_unlock( &p->mutex);
        return;
    }

    p->wait_for_me++;
    pthread_mutex_unlock( &p->mutex);

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

    pthread_mutex_lock( &lsredis_mutex);
    while( lsredis_running == 0)
        pthread_cond_wait( &lsredis_cond, &lsredis_mutex);

    redisAsyncCommand( wrac, NULL, NULL, "MULTI");
    redisAsyncCommandArgv( wrac, NULL, NULL, 4, (const char **)argv, NULL);

    redisAsyncCommand( wrac, NULL, NULL, "PUBLISH %s %s", lsredis_publisher
        , p->key);
    redisAsyncCommand( wrac, NULL, NULL, "EXEC");
    pthread_mutex_unlock( &lsredis_mutex);

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

```

7.7.2.28 void lsredis_sig_service (struct pollfd * evt)

Parameters

in	evt	The pollfd object that triggered this call
----	-----	--

Definition at line 913 of file lsredis.c.


```

    {
        struct signalfd_siginfo fdsi;

        //
        // Really, we don't care about the signal,
        // it's just used to drop out of the poll
        // function when there is something for us
        // to do.
        //

        read( evt->fd, &fdsi, sizeof( struct signalfd_siginfo));
    }

```

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

Use the publication to request the new value.

Definition at line 635 of file lsredis.c.

```

{
    redisReply *r;
    lsredis_obj_t *p;
    char *k;
    char *publisher;
    ENTRY htab_input, *htab_output;
    int err;

    r = (redisReply *)reply;

    // Ignore our psubscribe reply
    //
    if( r->type == REDIS_REPLY_ARRAY && r->elements == 3 && r->element[0]->type
        == REDIS_REPLY_STRING && strcmp( r->element[0]->str, "psubscribe")==0)
        return;

    // But log other stuff we don't understand
    //
    if( r->type != REDIS_REPLY_ARRAY ||
        r->elements != 4 ||
        r->element[3]->type != REDIS_REPLY_STRING ||
        r->element[2]->type != REDIS_REPLY_STRING) {

        lslogging_log_message( "lsredis_subCB: unexpected
            reply");
        lsredis_debugCB( ac, reply, privdata);
        return;
    }

    //
    // Ignore obvious junk
    //
    k = r->element[3]->str;

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

    //
    // see if we care
    //
    if( regexexec( &lsredis_key_select_regex, k, 0, NULL, 0
        ) == 0) {
        //
        // We should know about this one
        //

        htab_input.key = k;
        htab_input.data = NULL;

        errno = 0;
        err = hsearch_r( htab_input, FIND, &htab_output, &lsredis_htab)
            ;
        if( err == 0 && errno == ESRCH)
            p = NULL;
        else
            p = htab_output->data;

        if( p == NULL) {
            _lsredis_get_obj( k);
        } else {

```

```

    // Look who's talk'n
    publisher = r->element[2]->str;

    pthread_mutex_lock( &p->mutex);
    if( p->wait_for_me) {
        //
        // see if we are done waiting
        //
        if( strcmp( publisher, lsredis_publisher)==0)
            p->wait_for_me--;

        pthread_mutex_unlock( &p->mutex);
        //
        // Don't get a new value, either we set it last or we are still waiting
        // for redis to report
        // our publication
        //
        return;
    }

    // Here we know our value is out of date
    //
    p->valid = 0;
    lsevents_send_event( "%s Invalid", p->events_name
    );
    pthread_mutex_unlock( &p->mutex);

    //
    // We shouldn't get here if wait_for_me is zero and we are the publisher.
    // If somehow we did (ie we did an hset with out incrementing wait_for_me
    // or if we published too many times), it shouldn't hurt to get the value again.
    //
    redisAsyncCommand( roac, lsredis_hgetCB, p, "HGET %s
    VALUE", k);
}
}
}

```

7.7.2.30 void* lsredis_worker (void * dummy)

subscribe to changes and service sockets

< poll timeout, in millisecs (of course)

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

< number of active elements in fda

Definition at line 932 of file lsredis.c.

```

{
    static int poll_timeout_ms = -1;
    static struct pollfd fda[4];
    static int nfda = 0;
    static sigset_t our_sigset;
    int pollrtn;
    int i;

    pthread_mutex_lock( &lsredis_mutex);
    //
    // block ordinary signal mechanism
    //
    sigemptyset( &our_sigset);
    sigaddset( &our_sigset, SIGUSR1);
    pthread_sigmask( SIG_BLOCK, &our_sigset, NULL);

    // Set up fd mechanism
    //
    fda[0].fd = signalfd( -1, &our_sigset, SFD_NONBLOCK);
    if( fda[0].fd == -1) {
        char *es;

        es = strerror( errno);
        lslogging_log_message( "lsredis_worker: Signalfd
        trouble '%s'", es);
    }
    fda[0].events = POLLIN;
    nfda = 1;

    lsredis_running = 1;
}

```

```

if( redisAsyncCommand( subac, lsredis_subCB, NULL, "
    PSUBSCRIBE REDIS_KV_CONNECTOR UI* MD2-*" ) == REDIS_ERR) {
    lslogging_log_message( "Error sending PSUBSCRIBE
        command");
}

redisAsyncCommand( roac, lsredis_keysCB, NULL, "KEYS *");
pthread_cond_signal( &lsredis_cond);
pthread_mutex_unlock( &lsredis_mutex);

while(1) {
    nfda = 1;

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

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

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

    pollrtn = poll( fda, nfda, poll_timeout_ms);

    if( pollrtn && fda[0].revents) {
        lsredis_sig_service( &(fda[0]));
        pollrtn--;
    }

    for( i=1; i<nfda; i++) {
        if( fda[i].revents) {
            lsredis_fd_service( &(fda[i]));
        }
    }
}
}

```

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

call back in case a redis server becomes disconnected TODO: reconnect

Definition at line 560 of file Lsredis.c.

```

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

```

7.7.3 Variable Documentation

7.7.3.1 pthread_cond_t lsredis_cond [static]

Definition at line 75 of file Lsredis.c.

7.7.3.2 char* lsredis_head = NULL [static]

Definition at line 88 of file Lsredis.c.

7.7.3.3 `struct hsearch_data lsredis_hstab` `[static]`

Definition at line 80 of file lsredis.c.

7.7.3.4 `regex_t lsredis_key_select_regex` `[static]`

Definition at line 87 of file lsredis.c.

7.7.3.5 `pthread_mutex_t lsredis_mutex = PTHREAD_RECURSIVE_MUTEX_INITIALIZER_NP` `[static]`

Definition at line 73 of file lsredis.c.

7.7.3.6 `lsredis_obj_t* lsredis_objs = NULL` `[static]`

Definition at line 79 of file lsredis.c.

7.7.3.7 `char* lsredis_publisher = NULL` `[static]`

Definition at line 86 of file lsredis.c.

7.7.3.8 `int lsredis_running = 0` `[static]`

Definition at line 76 of file lsredis.c.

7.7.3.9 `pthread_t lsredis_thread` `[static]`

Definition at line 71 of file lsredis.c.

7.7.3.10 `redisAsyncContext* roac` `[static]`

Definition at line 83 of file lsredis.c.

7.7.3.11 `struct pollfd rofd` `[static]`

Definition at line 91 of file lsredis.c.

7.7.3.12 `redisAsyncContext* subac` `[static]`

Definition at line 82 of file lsredis.c.

7.7.3.13 `struct pollfd subfd` `[static]`

Definition at line 90 of file lsredis.c.

7.7.3.14 `redisAsyncContext* wrac` `[static]`

Definition at line 84 of file lsredis.c.

7.7.3.15 struct pollfd wrfd [static]

Definition at line 92 of file lsredis.c.

7.8 ltimer.c File Reference

Support for delayed and periodic events.

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

Data Structures

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

Macros

- #define [LSTIMER_LIST_LENGTH](#) 1024
We'll allow this many timers. This should be way more than enough.
- #define [LSTIMER_RESOLUTION_NSECS](#) 100000
times within this amount in the future are considered "now" and the events should be called

Typedefs

- typedef struct [ltimer_list_struct](#) [ltimer_list_t](#)
Everything we need to know about a timer.

Functions

- void [ltimer_add_timer](#) (char *event, int shots, unsigned long int secs, unsigned long int nsecs)
Create a timer.
- static void [service_timers](#) ()
Send events that are past due, due, or just about to be due.
- static void [handler](#) (int sig, siginfo_t *si, void *dummy)
Service the signal.
- static void * [ltimer_worker](#) (void *dummy)
Our worker.
- void [ltimer_init](#) ()
Initialize the timer list and pthread stuff.
- void [ltimer_run](#) ()
Start up our thread.

Variables

- static int [ltimer_active_timers](#) = 0
count of the number timers we are tracking
- static [ltimer_list_t](#) [ltimer_list](#) [[LSTIMER_LIST_LENGTH](#)]
Our timer list.
- static pthread_t [ltimer_thread](#)

- the timer thread*
- static pthread_mutex_t [lstimer_mutex](#)
protect the timer list
- static pthread_cond_t [lstimer_cond](#)
allows us to be idle when there is nothing to do
- static timer_t [lstimer_timerid](#)
our real time timer
- static int [new_timer](#) = 0
indicate that a new timer exists and a call to service_timers is required

7.8.1 Detailed Description

Support for delayed and periodic events.

Date

2012

Author

Keith Brister

Copyright

All Rights Reserved

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

7.8.2 Macro Definition Documentation

7.8.2.1 #define LSTIMER_LIST_LENGTH 1024

We'll allow this many timers. This should be way more than enough.

Definition at line 11 of file [lstimer.c](#).

7.8.2.2 #define LSTIMER_RESOLUTION_NSECS 100000

times within this amount in the future are considered "now" and the events should be called

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

7.8.3 Typedef Documentation

7.8.3.1 typedef struct lstimer_list_struct lstimer_list_t

Everything we need to know about a timer.

7.8.4 Function Documentation

7.8.4.1 static void handler (int sig, siginfo_t * si, void * dummy) [static]

Service the signal.

Definition at line 174 of file [lstimer.c](#).

```

    {
        pthread_mutex_lock( &lstimer_mutex);
        service_timers();
        pthread_mutex_unlock( &lstimer_mutex);
    }

```

7.8.4.2 void lstimer_add_timer(char * event, int shots, unsigned long int secs, unsigned long int nsecs)

Create a timer.

Parameters

<i>event</i>	Name of the event to send when the timer goes off
<i>shots</i>	Number of times to run. 0 means never, -1 means forever
<i>secs</i>	Number of seconds to wait
<i>nsecs</i>	Number of nano-seconds to run in addition to secs

Definition at line 50 of file lstimer.c.

```

    {
        int i;
        struct timespec now;

        // Time we were called. Delay is based on call time, not queued time
        //
        clock_gettime( CLOCK_REALTIME, &now);

        pthread_mutex_lock( &lstimer_mutex);

        for( i=0; i<LSTIMER_LIST_LENGTH; i++) {
            if( lstimer_list[i].shots == 0)
                break;
        }

        if( i == LSTIMER_LIST_LENGTH) {
            pthread_mutex_unlock( &lstimer_mutex);

            lslogging_log_message( "lstimer_add_timer: out of
                timers for event: %s, shots: %d, secs: %u, nsecs: %u",
                event, shots, secs, nsecs);

            return;
        }

        strncpy( lstimer_list[i].event, event, LSEVENTS_EVENT_LENGTH
            - 1);
        lstimer_list[i].event[LSEVENTS_EVENT_LENGTH
            - 1] = 0;
        lstimer_list[i].shots      = shots;
        lstimer_list[i].delay_secs = secs;
        lstimer_list[i].delay_nsecs = nsecs;

        lstimer_list[i].next_secs = secs + now.tv_sec + (
            now.tv_nsec + nsecs) / 1000000000;
        lstimer_list[i].next_nsecs = (now.tv_nsec + nsecs
            ) % 1000000000;
        lstimer_list[i].last_secs = 0;
        lstimer_list[i].last_nsecs = 0;

        lstimer_list[i].ncalls = 0;
        lstimer_list[i].init_secs = now.tv_sec;
        lstimer_list[i].init_nsecs = now.tv_nsec;

        if( shots != 0) {
            lstimer_active_timers++;
            new_timer++;
        }

        pthread_cond_signal( &lstimer_cond);
        pthread_mutex_unlock( &lstimer_mutex);
    }

```

7.8.4.3 void lstimer_init ()

Initialize the timer list and pthread stuff.

Definition at line 259 of file lstimer.c.

```

    {
        int i;

        for( i=0; i<LSTIMER_LIST_LENGTH; i++) {
            lstimer_list[i].shots = 0;
        }

        pthread_mutex_init( &lstimer_mutex, NULL);
        pthread_cond_init( &lstimer_cond, NULL);
    }

```

7.8.4.4 void lstimer_run ()

Start up our thread.

Definition at line 273 of file lstimer.c.

```

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

```

7.8.4.5 static void* lstimer_worker (void * dummy) [static]

Our worker.

The main loop runs when a new timer is added. The service routine deals with maintenance.

Parameters

in	<i>dummy</i>	required by protocol
----	--------------	----------------------

Definition at line 184 of file lstimer.c.

```

    {
        int
            known_timers;

        struct sigevent sev;
        struct sigaction sa;
        sigset_t mask;

        // See example at
        // http://www.kernel.org/doc/man-pages/online/pages/man2/timer_create.2.html
        //

        // Set up handler
        //
        sa.sa_flags = SA_SIGINFO;
        sa.sa_sigaction = handler;
        sigemptyset(&sa.sa_mask);
        if (sigaction(SIGRTMIN, &sa, NULL) == -1) {
            lslogging_log_message( "lstimer_worker: sigaction
                failed");
            exit( -1);
        }

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

```



```

// Block timer signal for now since we really
// want to be sure we do not own a lock on the timer mutex
// while servicing the signal
//
sigemptyset( &mask);
sigaddset( &mask, SIGRTMIN);

known_timers = 0;

while( 1) {
    pthread_mutex_lock( &lstmimer_mutex);

    while( new_timer == 0)
        pthread_cond_wait( &lstmimer_cond, &lstmimer_mutex
        );

    // ignore signals so we don't service the signal while we are already in
    // the
    // service routine
    //
    sigprocmask( SIG_SETMASK, &mask, NULL);

    //
    // Setting up the timer interval is in the handler
    // so just call it
    //
    service_timers();

    //
    // Reset our flag
    //
    new_timer = 0;

    pthread_mutex_unlock( &lstmimer_mutex);

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

```

7.8.4.6 static void service_timers () [static]

Send events that are past due, due, or just about to be due.

Definition at line 102 of file ltimer.c.

```

{
    int
    i,
    found_active;

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

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

    // Call with lstmimer_mutex locked

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

    found_active = 0;
    for( i=0; i<lstmimer_active_timers; i++) {
        p = &(lstmimer_list[i]);
        if( p->shots != 0) {
            found_active++;
            if( p->next_secs < then.tv_sec || (p->next_secs ==
            then.tv_sec && p->next_nsec <= then.tv_nsec)) {

```

```

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

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

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

```

7.8.5 Variable Documentation

7.8.5.1 `int lstimer_active_timers = 0` [static]

count of the number timers we are tracking

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

7.8.5.2 `pthread_cond_t lstimer_cond` [static]

allows us to be idle when there is nothing to do

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

7.8.5.3 `lstimer_list_t lstimer_list[LSTIMER_LIST_LENGTH]` [static]

Our timer list.

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

7.8.5.4 `pthread_mutex_t lstimer_mutex` [static]

protect the timer list

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

7.8.5.5 pthread_t ltimer_thread [static]

the timer thread

Definition at line 38 of file ltimer.c.

7.8.5.6 timer_t ltimer_timerid [static]

our real time timer

Definition at line 41 of file ltimer.c.

7.8.5.7 int new_timer = 0 [static]

indicate that a new timer exists and a call to service_timers is required

Definition at line 42 of file ltimer.c.

7.9 md2cmds.c File Reference

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

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

Functions

- void [md2cmds_home_prep](#) ()
- int [md2cmds_home_wait](#) (double timeout_secs)
- void [md2cmds_move_prep](#) ()
prepare for new movements
- int [md2cmds_move_wait](#) (double timeout_secs)
Wait for all the motions requested to complete.
- int [md2cmds_is_moving](#) ()
returns non-zero if we think a motor is moving, 0 otherwise
- double [md2cmds_prep_axis](#) (lspmac_motor_t *mp, double pos)
- void [md2cmds_organs_move_presets](#) (char *pay, char *paz, char *pcy, char *pcz, char *psz)
- void [md2cmds_transfer](#) ()
Transfer a sample.
- int [md2cmds_moveAbs](#) (const char *ccmd)
Move a motor to the position requested.
- void [md2cmds_phase_change](#) (const char *ccmd)
Move md2 devices to a preconfigured state.
- void [md2cmds_mvcenter_move](#) (double cx, double cy, double ax, double ay, double az)
Move the centering and alignment tables.
- void [md2cmds_maybe_done_moving_cb](#) (char *event)
Track how many motors are moving.
- void [md2cmds_maybe_done_homing_cb](#) (char *event)
Track motors homing.
- void [md2cmds_kappaphi_move](#) (double kappa_deg, double phi_deg)
- 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_time_capz_cb` (char *event)
- *Time the capillary motion for the transfer routine.*
- int `md2cmds_action_queue` (double timeout, char *action)
- void `md2cmds_abort` ()
- void `md2cmds_action_wait` ()
- *pause until md2cmds_worker has finished running the command*
- void * `md2cmds_worker` (void *dummy)
- *Our worker thread.*
- void `md2cmds_coordsys_1_stopped_cb` (char *event)
- void `md2cmds_coordsys_2_stopped_cb` (char *event)
- void `md2cmds_coordsys_3_stopped_cb` (char *event)
- void `md2cmds_coordsys_4_stopped_cb` (char *event)
- void `md2cmds_coordsys_5_stopped_cb` (char *event)
- void `md2cmds_coordsys_7_stopped_cb` (char *event)
- void `md2cmds_init` ()
- *Initialize the md2cmds module.*
- void `md2cmds_run` ()
- *Start up the thread.*

Variables

- pthread_cond_t `md2cmds_cond`
- *condition to signal when it's time to run an md2 command*
- pthread_mutex_t `md2cmds_mutex`
- *mutex for the condition*
- int `md2cmds_moving_queue_wait` = 0
- pthread_cond_t `md2cmds_moving_cond`
- *wait for command to have been dequeued and run*
- pthread_mutex_t `md2cmds_moving_mutex`
- *message passing between md2cmds and pg*
- int `md2cmds_homing_count` = 0
- *We've asked a motor to home.*
- pthread_cond_t `md2cmds_homing_cond`
- *coordinate homing and homed*
- pthread_mutex_t `md2cmds_homing_mutex`
- *our mutex;*
- int `md2cmds_moving_count` = 0
- char `md2cmds_cmd` [MD2CMDS_CMD_LENGTH]
- *our command;*
- `lsredis_obj_t * md2cmds_md_status_code`
- static pthread_t `md2cmds_thread`
- static int `rotating` = 0
- *flag: when omega is in position after a rotate we want to re-home omega*
- static double `md2cmds_capz_moving_time` = NAN

7.9.1 Detailed Description

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

Date

2012

Author

Keith Brister

Copyright

All Rights Reserved

Definition in file [md2cmds.c](#).

7.9.2 Function Documentation

7.9.2.1 void md2cmds_abort ()

Definition at line 1270 of file md2cmds.c.

```

{
    //
    // First priority is to close the shutter
    //
    if( fshut->moveAbs( fshut, 0))
        lslogging_log_message( "md2cmds_abort: for some reason
            the shutter close requested failed. Proceeding anyway.");

    //
    // Now stop all the motors
    //
    lspmac_abort();
    if( md2cmds_move_wait( 10.0))
        lslogging_log_message( "md2cmds_abort: Some motors did
            not appear to stop. Proceeding with reset anyway");

    //
    // Now try to close the shutter (again)
    //
    if( fshut->moveAbs( fshut, 0))
        lslogging_log_message( "md2cmds_abort: for some reason
            the shutter close requested failed (2). Proceeding anyway.");

    //
    // Force the motion flags down
    //
    lspmac_SockSendDPLine( NULL, "m5075=0");
}

```

7.9.2.2 int md2cmds_action_queue (double timeout, char * action)

Definition at line 1235 of file md2cmds.c.

```

{
    int rtn;
    struct timespec waitforit;

    if( timeout < 0.0) {
        rtn = pthread_mutex_lock( &md2cmds_mutex);
    } else {
        clock_gettime( CLOCK_REALTIME, &waitforit);
    }
}

```

```

    waitforit.tv_sec += floor(timeout);

    waitforit.tv_nsec += (timeout - waitforit.tv_sec)*1.e9;
    while( waitforit.tv_nsec >= 1000000000) {
        waitforit.tv_sec++;
        waitforit.tv_nsec -= 1000000000;
    }

    rtn = pthread_mutex_timedlock( &md2cmds_mutex, &waitforit);
}

if( rtn == 0) {
    strncpy( md2cmds_cmd, action, MD2CMDS_CMD_LENGTH
        -1);
    md2cmds_cmd[MD2CMDS_CMD_LENGTH-1] = 0;
    pthread_cond_signal( &md2cmds_cond);
    pthread_mutex_unlock( &md2cmds_mutex);
} else {
    if( rtn == ETIMEDOUT)
        lslogging_log_message( "md2cmds_action_queue: %s not
            queued, operation timed out", action);
    else
        lslogging_log_message( "md2cmds_action_queue: %s not
            queued with error code %d", action, rtn);
}
return rtn;
}

```

7.9.2.3 void md2cmds_action_wait ()

pause until md2cmds_worker has finished running the command

Definition at line 1300 of file md2cmds.c.

```

{
    pthread_mutex_lock( &md2cmds_mutex);
    pthread_mutex_unlock( &md2cmds_mutex);
}

```

7.9.2.4 void md2cmds_center ()

Move centering and alignment tables as requested TODO: Implement.

Definition at line 1199 of file md2cmds.c.

```

{
}

```

7.9.2.5 void md2cmds_collect ()

Collect some data.

- < index of shot to be taken
- < Exposure time (saved to compute shutter timeout)
- < start cnts
- < delta cnts
- < omega velocity cnts/msec
- < acceleration time (msec)
- < exposure time (msec)
- < one of the stages, at least, needs to be moved
- < unit to counts conversion

- < nominal zero offset
- < maximum acceleration allowed for omega
- < current kappa position in case we need to move phi only
- < current phi position in case we need to move kappa only
- < setup timeouts for shutter

Definition at line 779 of file md2cmds.c.

```

    {
long long skey;
double exp_time;
double p170;
double p171;
double p173;
double p175;
double p180;
int center_request;
double u2c;
double neutral_pos;
double max_accel;
double kappa_pos;
double phi_pos;
struct timespec now, timeout;
int err;

u2c      = lsredis_getd( omega->u2c);
neutral_pos = lsredis_getd( omega->neutral_pos);
max_accel  = lsredis_getd( omega->max_accel);

md2cmds_move_prep();
md2cmds_organs_move_presets( "In", "In", "In", "In",
    "Cover");
//
if( md2cmds_move_wait( 30.0)) {
    lslogging_log_message( "md2cmds_collect: Timed out
        waiting for organs to move. Aborting data collection.");
    lsevents_send_event( "Data Collection Aborted");
    return;
}

//
// reset shutter has opened flag
//
lspmac_SockSendDPlane( NULL, "P3001=0 P3002=0");

while( 1) {
    lspg_nextshot_call();
    lspg_nextshot_wait();

    exp_time = lspg_nextshot.dsexp;

    if( lspg_nextshot.no_rows_returned) {
        lspg_nextshot_done();
        break;
    }

    skey = lspg_nextshot.skey;
    lspg_query_push( NULL, "SELECT px.shots_set_state(%lld,
        'Preparing')", skey);

    center_request = 0;
    if( lspg_nextshot.active) {
        if(
            //
            // Don't move if we are within 0.1 microns of our destination
            //
            (fabs( lspg_nextshot.cx - cenx->position) >
            0.1) ||
            (fabs( lspg_nextshot.cy - ceny->position) >
            0.1) ||
            (fabs( lspg_nextshot.ax - alignx->position
            ) > 0.1) ||
            (fabs( lspg_nextshot.ay - aligny->position
            ) > 0.1) ||
            (fabs( lspg_nextshot.az - alignz->position
            ) > 0.1)) {

                center_request = 1;
                lslogging_log_message( "md2cmds_collect: moving
                    center to cx=%f, cy=%f, ax=%f, ay=%f, az=%f",lspg_nextshot.cx,
                    lspg_nextshot.cy, lspg_nextshot.ax, lspg_nextshot

```

```

        .ay, lspg_nextshot.az);
        md2cmds_move_prep();
        md2cmds_mvcenter_move( lspg_nextshot.
cx, lspg_nextshot.cy, lspg_nextshot.ax,
lspg_nextshot.ay, lspg_nextshot.az);
        if( md2cmds_move_wait( 2.0)) {
            lslogging_log_message( "md2cmds_collect: Timed
out waiting for alignment or centering stage (or both) to stop moving.  Aborting
data collection.");
            lsevents_send_event( "Data Collection Aborted");
            return;
        }
    }
}

// Maybe move kappa and/or phi
//
if( !lspg_nextshot.dsphi_isnull || !lspg_nextshot
.dskappa_isnull) {

    kappa_pos = lspg_nextshot.dskappa_isnull ?
    lspmac_getPosition( kappa) : lspg_nextshot.
    dskappa;
    phi_pos = lspg_nextshot.dsphi_isnull ?
    lspmac_getPosition( phi) : lspg_nextshot.
    dsphi;

    lslogging_log_message( "md2cmds_collect: move
phy/kappa: kappa=%f phi=%f", kappa_pos, phi_pos);
    md2cmds_move_prep();
    md2cmds_kappaphi_move( kappa_pos, phi_pos);
    if( md2cmds_move_wait( 30.0)) {
        lslogging_log_message( "md2cmds_collect: Timed
out waiting for kappa or phi (or both) to stop moving.  Aborting data collection.
");
        lsevents_send_event( "Data Collection Aborted");
        return;
    }
}

//
// Calculate the parameters we'll need to run the scan
//
p180 = lspg_nextshot.dsexp * 1000.0;
p170 = u2c * (lspg_nextshot.sstart + neutral_pos);
p171 = u2c * lspg_nextshot.dsowidth;
p173 = fabs(p180) < 1.e-4 ? 0.0 : u2c * lspg_nextshot.dsowidth
/ p180;
p175 = p173/max_accel;

//
// free up access to nextshot
//
lspg_nextshot_done();

//
// prepare the database and detector to expose
// On exit we own the diffractometer lock and
// have checked that all is OK with the detector
//
lspg_seq_run_prep_all( skey,
                      kappa->position,
                      phi->position,
                      cenx->position,
                      ceny->position,
                      alignx->position,
                      aligny->position,
                      alignz->position
                      );

//
// make sure our opened flag is down
// wait for the p3001=0 command to be noticed
//
clock_gettime( CLOCK_REALTIME, &now);
timeout.tv_sec = now.tv_sec + 10;
timeout.tv_nsec = now.tv_nsec;

err = 0;
pthread_mutex_lock( &lspmac_shutter_mutex);
while( err == 0 && lspmac_shutter_has_opened == 1)
    err = pthread_cond_timedwait( &lspmac_shutter_cond, &
    lspmac_shutter_mutex, &timeout);
pthread_mutex_unlock( &lspmac_shutter_mutex);

```



```

if( err == ETIMEDOUT) {
    pthread_mutex_unlock( &lspmac_shutter_mutex);
    lslogging_log_message( "md2cmds_collect: Timed out
    waiting for shutter to open. Data collection aborted.");
    lsevents_send_event( "Data Collection Aborted");
    return;
}

//
// Start the exposure
//
md2cmds_move_prep();
lspmac_SockSendDpline( "Exposure", "&1 P170=%.1f
    P171=%.1f P173=%.1f P174=0 P175=%.1f P176=0 P177=1 P178=0 P180=%.1f M431=1 &1B131R",
    p170,      p171,      p173,      p175,
    p180);

//
// We could look for the "Exposure command accepted" event at this point.
//
//
// wait for the shutter to open
//
clock_gettime( CLOCK_REALTIME, &now);
timeout.tv_sec = now.tv_sec + 10;
timeout.tv_nsec = now.tv_nsec;

err = 0;
pthread_mutex_lock( &lspmac_shutter_mutex);
while( err == 0 && lspmac_shutter_has_opened == 0)
    err = pthread_cond_timedwait( &lspmac_shutter_cond, &
    lspmac_shutter_mutex, &timeout);

if( err == ETIMEDOUT) {
    pthread_mutex_unlock( &lspmac_shutter_mutex);
    lslogging_log_message( "md2cmds_collect: Timed out
    waiting for shutter to open. Data collection aborted.");
    lsevents_send_event( "Data Collection Aborted");
    return;
}

//
// wait for the shutter to close
//
clock_gettime( CLOCK_REALTIME, &now);
timeout.tv_sec = now.tv_sec + 4 + exp_time; // hopefully 4 seconds
    is long enough to never catch a legitimate shutter close and short enough to
    bail when something is really wrong
timeout.tv_nsec = now.tv_nsec;

err = 0;
while( err == 0 && lspmac_shutter_state == 1)
    err = pthread_cond_timedwait( &lspmac_shutter_cond, &
    lspmac_shutter_mutex, &timeout);
pthread_mutex_unlock( &lspmac_shutter_mutex);

if( err == ETIMEDOUT) {
    pthread_mutex_unlock( &lspmac_shutter_mutex);
    lslogging_log_message( "md2cmds_collect: Timed out
    waiting for shutter to close. Data collection aborted.");
    lsevents_send_event( "Data Collection Aborted");
    return;
}

//
// Signal the detector to start reading out
//
lspg_query_push( NULL, "SELECT px.unlock_diffractionmeter()");

//
// Update the shot status
//
lspg_query_push( NULL, "SELECT px.shots_set_state(%lld,
    'Writing')", skey);

//
// reset shutter has opened flag
//
lspmac_SockSendDpline( NULL, "P3001=0");

//
// Wait for omega to stop moving
//

```

```

if( md2cmds_move_wait( 10.0) ) {
    lslogging_log_message( "md2cmds_collect: Giving up
        waiting for omega to stop moving. Data collection aborted.");
    lsevents_send_event( "Data Collection Aborted");
    return;
}

//
// Move the center/alignment stages to the next position
//
// TODO: position omega for the next shot. During data collection the
// motion program
// makes a good guess but for ortho snaps it is wrong. We should add an
// argument to the motion program
//

if( !lspg_nextshot.active2_isnull &&
    lspg_nextshot.active2 ) {
    if(
        (fabs( lspg_nextshot.cx2 - cenx->position)
        > 0.1) ||
        (fabs( lspg_nextshot.cy2 - ceny->position)
        > 0.1) ||
        (fabs( lspg_nextshot.ax2 - alignx->position)
        > 0.1) ||
        (fabs( lspg_nextshot.ay2 - aligny->position)
        > 0.1) ||
        (fabs( lspg_nextshot.az2 - alignz->position)
        > 0.1)) {

        center_request = 1;
        md2cmds_move_prep();
        md2cmds_mvcenter_move( lspg_nextshot.
            cx, lspg_nextshot.cy, lspg_nextshot.ax,
            lspg_nextshot.ay, lspg_nextshot.az);
    }
}
lsevents_send_event( "Data Collection Done");
}

```

7.9.2.6 void md2cmds.coordsys.1_stopped_cb (char * event)

Definition at line 1340 of file md2cmds.c.

```

{
}

```

7.9.2.7 void md2cmds.coordsys.2_stopped_cb (char * event)

Definition at line 1342 of file md2cmds.c.

```

{
}

```

7.9.2.8 void md2cmds.coordsys.3_stopped_cb (char * event)

Definition at line 1344 of file md2cmds.c.

```

{
}

```

7.9.2.9 void md2cmds.coordsys.4_stopped_cb (char * event)

Definition at line 1346 of file md2cmds.c.

```

{
}

```

7.9.2.10 void md2cmds_coordsys_5_stopped_cb (char * event)

Definition at line 1348 of file md2cmds.c.

```

{
}

```

7.9.2.11 void md2cmds_coordsys_7_stopped_cb (char * event)

Definition at line 1350 of file md2cmds.c.

```

{
}

```

7.9.2.12 void md2cmds_home_prep ()

Definition at line 34 of file md2cmds.c.

```

{
pthread_mutex_lock( &md2cmds_homing_mutex);
md2cmds_homing_count = -1;
pthread_mutex_unlock( &md2cmds_homing_mutex);
}

```

7.9.2.13 int md2cmds_home_wait (double timeout_secs)

Definition at line 41 of file md2cmds.c.

```

{
struct timespec timeout, now;
double isecs, fsecs;
int err;

clock_gettime( CLOCK_REALTIME, &now);

fsecs = modf( timeout_secs, &isecs);

timeout.tv_sec = now.tv_sec + (long)floor( isecs);
timeout.tv_nsec = now.tv_nsec + (long)floor( fsecs * 1.0e9);

timeout.tv_sec += timeout.tv_nsec / 1000000000;
timeout.tv_nsec %= 1000000000;

err = 0;
pthread_mutex_lock( &md2cmds_homing_mutex);
while( err == 0 && md2cmds_homing_count == -1)
    err = pthread_cond_timedwait( &md2cmds_homing_cond, &
        md2cmds_homing_mutex, &timeout);

if( err != 0) {
    if( err != ETIMEDOUT) {
        lslogging_log_message( "md2cmds_home_wait:
            unexpected error from timedwait: %d tv_sec %ld tv_nsec %ld", err, timeout.tv_sec,
            timeout.tv_nsec);
    }
    pthread_mutex_unlock( &md2cmds_homing_mutex);
    return 1;
}

err = 0;
while( err == 0 && md2cmds_homing_count > 0)
    err = pthread_cond_timedwait( &md2cmds_homing_cond, &
        md2cmds_homing_mutex, &timeout);
pthread_mutex_unlock( &md2cmds_homing_mutex);

if( err != 0) {
    if( err != ETIMEDOUT)
        lslogging_log_message( "md2cmds_home_wait:
            unexpected error from timedwait: %d", err);
}
}

```

```

    return 1;
}
return 0;
}

```

7.9.2.14 void md2cmds_init ()

Initialize the md2cmds module.

Definition at line 1356 of file md2cmds.c.

```

{
pthread_mutexattr_t mutex_initializer;

pthread_mutexattr_init( &mutex_initializer);
pthread_mutexattr_settype( &mutex_initializer, PTHREAD_MUTEX_RECURSIVE);

pthread_mutex_init( &md2cmds_mutex, &mutex_initializer);
pthread_cond_init( &md2cmds_cond, NULL);

pthread_mutex_init( &md2cmds_moving_mutex, &
    mutex_initializer);
pthread_cond_init( &md2cmds_moving_cond, NULL);

pthread_mutex_init( &md2cmds_homing_mutex, &
    mutex_initializer);
pthread_cond_init( &md2cmds_homing_cond, NULL);

md2cmds_md_status_code = lsredis_get_obj
( "md2_status_code");
lsredis_setstr( md2cmds_md_status_code, "
7");
}

```

7.9.2.15 int md2cmds_is_moving ()

returns non-zero if we think a motor is moving, 0 otherwise

Definition at line 140 of file md2cmds.c.

```

{
int rtn;

pthread_mutex_lock( &md2cmds_moving_mutex);
rtn = md2cmds_moving_count != 0;
pthread_mutex_unlock( &md2cmds_moving_mutex);

return rtn;
}

```

7.9.2.16 void md2cmds_kappaphi_move (double kappa_deg, double phi_deg)

Definition at line 759 of file md2cmds.c.

```

{
int kc, pc;

// coordinate system 7
// 1 << (coord sys no - 1) = 64

kc = md2cmds_prep_axis( kappa, kappa_deg);
pc = md2cmds_prep_axis( kappa, phi_deg);

// ;150          LS-CAT Move X, Y Absolute
// ;           Q20   = X Value
// ;           Q21   = Y Value
// ;           Q100  = 1 << (coord sys no - 1)

lspmac_SockSendDPLine( "kappaphi_move", "&7 Q20=%d
Q21=%d Q100=64", kc, pc);
}

```

7.9.2.17 void md2cmds_maybe_done_homing_cb (char * event)

Track motors homing.

Definition at line 735 of file md2cmds.c.

```

{
pthread_mutex_lock( &md2cmds_homing_mutex);

if( strstr( event, "Homing") == NULL) {
    if( md2cmds_homing_count != -1)
        md2cmds_homing_count = 1;
    else
        md2cmds_homing_count++;
} else {
    if( md2cmds_homing_count > 0)
        md2cmds_homing_count--;
}

if( md2cmds_homing_count != 0)
    lsredis_setstr( md2cmds_md_status_code,
        "%s", "4");

if( md2cmds_homing_count == 0)
    pthread_cond_signal( &md2cmds_homing_cond);

pthread_mutex_unlock( &md2cmds_homing_mutex);
}

```

7.9.2.18 void md2cmds_maybe_done_moving_cb (char * event)

Track how many motors are moving.

Definition at line 706 of file md2cmds.c.

```

{

pthread_mutex_lock( &md2cmds_moving_mutex);
if( strstr( event, "Moving") != NULL) {
    //
    // -1 is a flag indicating we're expecting some action
    //
    if( md2cmds_moving_count == -1)
        md2cmds_moving_count = 1;
    else
        md2cmds_moving_count++;
} else {
    //
    //
    if( md2cmds_moving_count > 0)
        md2cmds_moving_count--;
}

lsredis_setstr( md2cmds_md_status_code, "
    %s", md2cmds_moving_count ? "4" : "3");

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

```

7.9.2.19 void md2cmds_maybe_rotate_done_cb (char * event)

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

Definition at line 1163 of file md2cmds.c.

```

{
    if( rotating) {
        rotating = 0;
        lsevents_send_event( "Rotate Done");
    }
}

```

7.9.2.20 void md2cmds_move_prep ()

prepare for new movements

Definition at line 86 of file md2cmds.c.

```

{
pthread_mutex_lock( &md2cmds_moving_mutex);
md2cmds_moving_count = -1;
pthread_mutex_unlock( &md2cmds_moving_mutex);
}

```

7.9.2.21 int md2cmds_move_wait (double timeout_secs)

Wait for all the motions requested to complete.

Parameters

<i>timeout_secs</i>	Double value of seconds to wait
---------------------	---------------------------------

There are two waits involved: First to wait for the first "Moving" to be seen and second to wait for the last "In Position". The timeout specified here is the sum of the two.

returns 0 on success and 1 if we timedout.

Definition at line 103 of file md2cmds.c.

```

{
double isecs, fsecs;
struct timespec timeout, now;
int err;

clock_gettime( CLOCK_REALTIME, &now);

fsecs = modf( timeout_secs, &isecs);

timeout.tv_sec = now.tv_sec + (long)floor( isecs);
timeout.tv_nsec = now.tv_nsec + (long)floor( fsecs * 1.0e9);

timeout.tv_sec += timeout.tv_nsec / 1000000000;
timeout.tv_nsec %= 1000000000;

err = 0;
pthread_mutex_lock( &md2cmds_moving_mutex);
while( err == 0 && md2cmds_moving_count == -1)
    err = pthread_cond_timedwait( &md2cmds_moving_cond, &
        md2cmds_moving_mutex, &timeout);

if( err == ETIMEDOUT) {
    pthread_mutex_unlock( &md2cmds_moving_mutex);
    return 1;
}

err = 0;
while( err == 0 && md2cmds_moving_count > 0)
    err = pthread_cond_timedwait( &md2cmds_moving_cond, &
        md2cmds_moving_mutex, &timeout);
pthread_mutex_unlock( &md2cmds_moving_mutex);

if( err == ETIMEDOUT)
    return 1;
return 0;
}

```

7.9.2.22 int 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 415 of file md2cmds.c.

```

    {
        char *cmd;
        char *ignore;
        char *ptr;
        char *mtr;
        char *pos;
        double fpos;
        char *endptr;
        lspmac_motor_t *mp;
        int i;
        int err;

        // ignore nothing
        if( ccmd == NULL || *ccmd == 0) {
            return 1;
        }

        // operate on a copy of the string since strtok_r will modify its argument
        //
        cmd = strdup( ccmd);

        // Parse the command string
        //
        ignore = strtok_r( cmd, " ", &ptr);
        if( ignore == NULL) {
            lslogging_log_message( "md2cmds_moveAbs: ignoring
                blank command '%s'", cmd);
            free( cmd);
            return 1;
        }

        // The first string should be "moveAbs" cause that's how we got here.
        // Toss it.

        mtr = strtok_r( NULL, " ", &ptr);
        if( mtr == NULL) {
            lslogging_log_message( "md2cmds moveAbs error: missing
                motor name");
            free( cmd);
            return 1;
        }

        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 1;
        }

        pos = strtok_r( NULL, " ", &ptr);
        if( pos == NULL) {
            lslogging_log_message( "md2cmds moveAbs error: missing
                position");
            free( cmd);
            return 1;
        }

        fpos = strtod( pos, &endptr);
        if( pos == endptr) {
            //
            // Maybe we have a preset. Give it a whirl
            // In any case we are done here.
            //
            err = lspmac_move_preset_queue( mp, pos);
            free( cmd);
            return err;
        }

        if( mp != NULL && mp->moveAbs != NULL) {
            wprintw( term_output, "Moving %s to %f\n", mtr, fpos);
            wnoutrefresh( term_output);
            err = mp->moveAbs( mp, fpos);
        }

        free( cmd);
        return err;
    }

```

7.9.2.23 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 679 of file md2cmds.c.

```

{

//
// centering stage is coordinate system 2
// alignment stage is coordinate system 3
//

double cx_cts, cy_cts, ax_cts, ay_cts, az_cts;

cx_cts = md2cmds_prep_axis( cenx, cx);
cy_cts = md2cmds_prep_axis( ceny, cy);
ax_cts = md2cmds_prep_axis( alignx, ax);
ay_cts = md2cmds_prep_axis( aligny, ay);
az_cts = md2cmds_prep_axis( alignz, az);

lspmac_SockSendDPLine( NULL, "&2 Q100=2 Q20=%.1f
Q21=%.1f B150R", cx_cts, cy_cts);
lspmac_SockSendDPLine( "mvcenter_move", "&3 Q100=4
Q30=%.1f Q31=%.1f Q32=%.1f B160R", ax_cts, ay_cts, az_cts);
}

```

7.9.2.24 void md2cmds_organs_move_presets (char * pay, char * paz, char * pcy, char * pcz, char * psz)

Definition at line 175 of file md2cmds.c.

```

{
double ay, az, cy, cz, sz;
int cay, caz, ccy, ccz, csz;
int err;

err = lsredis_find_preset( apery->name, pay, &ay)
;
if( err == 0) {
lslogging_log_message( "md2cmds_move_organs_presets:
no preset '%s' for motor '%s'", pay, apery->name);
return;
}

err = lsredis_find_preset( aperz->name, paz, &az)
;
if( err == 0) {
lslogging_log_message( "md2cmds_move_organs_presets:
no preset '%s' for motor '%s'", paz, aperz->name);
return;
}

err = lsredis_find_preset( capy->name, pcy, &cy);
if( err == 0) {
lslogging_log_message( "md2cmds_organs_move_presets:
no preset '%s' for motor '%s'", pcy, capy->name);
return;
}

err = lsredis_find_preset( capz->name, pcz, &cz);
if( err == 0) {
lslogging_log_message( "md2cmds_organs_move_presets:
no preset '%s' for motor '%s'", pcz, capz->name);
return;
}
}

```



```

err = lsredis_find_preset( scint->name, psz, &sz)
;
if( err == 0) {
    lslogging_log_message( "md2cmds_organs_move_presets:
        no preset '%s' for motor '%s'", psz, scint->name);
    return;
}

cay = md2cmds_prep_axis( apery, ay);
caz = md2cmds_prep_axis( aperz, az);
ccy = md2cmds_prep_axis( capy, cy);
ccz = md2cmds_prep_axis( capz, cz);
csz = md2cmds_prep_axis( scint, sz);

//
// 170          LS-CAT Move U, V, W, X, Y, Z Absolute
//              Q40      = X Value
//              Q41      = Y Value
//              Q42      = Z Value
//              Q43      = U Value
//              Q44      = V Value
//              Q45      = W Value
//
lspmac_SockSendDPLine( "organs", "&5 Q40=0 Q41=%d Q42=%d
    Q43=%d Q44=%d Q45=%d Q100=16 B170R", cay, caz, ccy, ccz, csz);
}

```

7.9.2.25 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 504 of file md2cmds.c.

```

{
char *cmd;
char *ignore;
char *ptr;
char *mode;
int err;

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) {
    lsevents_send_event( "Mode manualMount Starting");

    //
    // Try all motions, flag errors at the end
    //
}

```

```

md2cmds_move_prep();
err = lspmac_move_or_jog_preset_queue(
    kappa, "manualMount", 1);
err += lspmac_move_or_jog_preset_queue(
    omega, "manualMount", 0);
err += lspmac_move_or_jog_abs_queue( phi,
    0.0, 0);
err += lspmac_move_or_jog_preset_queue(
    aperz, "Cover", 1);
err += lspmac_move_or_jog_preset_queue( capz
    , "Cover", 1);
err += lspmac_move_or_jog_preset_queue(
    scint, "Cover", 1);
err += md2cmds_moveAbs( "moveAbs backLight 0");
err += md2cmds_moveAbs( "moveAbs backLight.intensity 0");
err += md2cmds_moveAbs( "moveAbs cryo 1");
err += md2cmds_moveAbs( "moveAbs fluo 0");
err += md2cmds_moveAbs( "moveAbs cam.zoom 1");

if( md2cmds_move_wait( 60.0) || err)
    lsevents_send_event( "Mode manualMount Aborted");
else
    lsevents_send_event( "Mode manualMount Done");
} else if( strcmp( mode, "robotMount") == 0) {

    lsevents_send_event( "Mode robotMount Starting");
    md2cmds_home_prep();
    md2cmds_move_prep();
    lspmac_homel_queue( kappa);
    lspmac_homel_queue( omega);
    lspmac_move_or_jog_abs_queue( phi, 0.0, 0);
    lspmac_move_or_jog_preset_queue( apery,
        "In", 1);
    lspmac_move_or_jog_preset_queue( aperz,
        "In", 1);
    lspmac_move_or_jog_preset_queue( capz,
        "Cover", 1);
    lspmac_move_or_jog_preset_queue( scint,
        "Cover", 1);
    md2cmds_moveAbs( "moveAbs backLight 0");
    md2cmds_moveAbs( "moveAbs backLight.intensity 0");
    md2cmds_moveAbs( "moveAbs cryo 1");
    md2cmds_moveAbs( "moveAbs fluo 0");
    md2cmds_moveAbs( "moveAbs cam.zoom 1");
    if( md2cmds_home_wait( 60.0)) {
        lsevents_send_event( "Mode robotMount Aborted");
    } else {
        if( md2cmds_move_wait( 60.0))
            lsevents_send_event( "Mode robotMount Aborted");
        else
            lsevents_send_event( "Mode robotMount Done");
    }
} else if( strcmp( mode, "center") == 0) {
    lsevents_send_event( "Mode center Starting");

    md2cmds_move_prep();
    md2cmds_moveAbs( "moveAbs kappa 0");
    md2cmds_moveAbs( "moveAbs omega 0");
    lspmac_move_or_jog_abs_queue( phi, 0.0,
        0);
    lspmac_move_or_jog_preset_queue( apery,
        "In", 1);
    lspmac_move_or_jog_preset_queue( aperz,
        "In", 1);
    lspmac_move_or_jog_preset_queue( capy,
        "In", 1);
    lspmac_move_or_jog_preset_queue( capz,
        "In", 1);
    lspmac_move_or_jog_preset_queue( scint,
        "Cover", 1);
    md2cmds_moveAbs( "moveAbs backLight 1");
    md2cmds_moveAbs( "moveAbs cam.zoom 1");
    md2cmds_moveAbs( "moveAbs cryo 0");
    md2cmds_moveAbs( "moveAbs fluo 0");

    if( md2cmds_move_wait( 60.0))
        lsevents_send_event( "Mode center Aborted");
    else
        lsevents_send_event( "Mode center Done");
} else if( strcmp( mode, "dataCollection") == 0) {

    lsevents_send_event( "Mode dataCollection Starting");

    md2cmds_move_prep();
    lspmac_move_or_jog_preset_queue( apery,

```

```

        "In", 1);
lspmac_move_or_jog_preset_queue( aperz,
        "In", 1);
lspmac_move_or_jog_preset_queue( capy,
        "In", 1);
lspmac_move_or_jog_preset_queue( capz,
        "In", 1);
lspmac_move_or_jog_preset_queue( scint,
        "Cover", 1);
md2cmds_moveAbs( "moveAbs backLight 0");
md2cmds_moveAbs( "moveAbs backLight.intensity 0");
md2cmds_moveAbs( "moveAbs cryo 0");
md2cmds_moveAbs( "moveAbs fluo 0");

if( md2cmds_move_wait( 60.0))
    lsevents_send_event( "Mode dataCollection Aborted");
else
    lsevents_send_event( "Mode dataCollection Done");
} else if( strcmp( mode, "beamLocation") == 0) {

    lsevents_send_event( "Mode beamLocation Starting");

    md2cmds_moveAbs( "moveAbs kappa 0");
    md2cmds_moveAbs( "moveAbs omega 0");
    lspmac_move_or_jog_preset_queue( apery,
        "In", 1);
    lspmac_move_or_jog_preset_queue( aperz,
        "In", 1);
    lspmac_move_or_jog_preset_queue( capy,
        "In", 1);
    lspmac_move_or_jog_preset_queue( capz,
        "In", 1);
    lspmac_move_or_jog_preset_queue( scint,
        "Scintillator", 1);
    md2cmds_moveAbs( "moveAbs backLight 0");
    md2cmds_moveAbs( "moveAbs cam.zoom 1");
    md2cmds_moveAbs( "moveAbs cryo 0");
    md2cmds_moveAbs( "moveAbs fluo 0");

    if( md2cmds_move_wait( 60.0))
        lsevents_send_event( "Mode beamLocation Aborted");
    else
        lsevents_send_event( "Mode beamLocation Done");
} else if( strcmp( mode, "safe") == 0) {

    lsevents_send_event( "Mode safe Starting");

    md2cmds_moveAbs( "moveAbs kappa 0");
    md2cmds_moveAbs( "moveAbs omega 0");
    lspmac_move_or_jog_preset_queue( apery,
        "In", 1);
    lspmac_move_or_jog_preset_queue( aperz,
        "Cover", 1);
    lspmac_move_or_jog_preset_queue( capy,
        "In", 1);
    lspmac_move_or_jog_preset_queue( capz,
        "Cover", 1);
    lspmac_move_or_jog_preset_queue( scint,
        "Cover", 1);
    md2cmds_moveAbs( "moveAbs backLight 0");
    md2cmds_moveAbs( "moveAbs cam.zoom 1");
    md2cmds_moveAbs( "moveAbs cryo 0");
    md2cmds_moveAbs( "moveAbs fluo 0");

    if( md2cmds_move_wait( 60.0))
        lsevents_send_event( "Mode safe Aborted");
    else
        lsevents_send_event( "Mode safe Done");
}

free( cmd);
}

```

7.9.2.26 double md2cmds_prep_axis (lspmac_motor_t * mp, double pos)

Definition at line 151 of file md2cmds.c.

```

{
    double rtn;
    double u2c;

```

```

double current_pos;
double neutral_pos;

pthread_mutex_lock( &(mp->mutex));

u2c      = lsredis_getd( mp->u2c);
neutral_pos = lsredis_getd( mp->neutral_pos);
current_pos = mp->position;

mp->motion_seen = 0;
mp->not_done    = 1;

rtn = u2c * (pos + neutral_pos);

pthread_mutex_unlock( &(mp->mutex));

return rtn;
}

```

7.9.2.27 void md2cmds_rotate ()

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

Definition at line 1026 of file md2cmds.c.

```

{
double cx, cy, ax, ay, az;
int mmask;

mmask = 0;
//
// BLUMax disables scintillator here.
//

//
// get the new center information
//
lspg_getcenter_call();
lspg_getcenter_wait();

// put up the back light
blight_ud->moveAbs( blight_ud, 1);

md2cmds_move_prep();
md2cmds_home_prep();

//
// make sure omega is homed
//
lspmac_homed_queue( omega);
//
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.dcx_isnull == 0 && fabs(
    lspg_getcenter.dcx) >= 0.0) ||
    (lspg_getcenter.dcy_isnull == 0 && fabs(
    lspg_getcenter.dcy) >= 0.0)) {
    mmask |= 2;
}

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;

if( (lspg_getcenter.dax_isnull == 0 && fabs(
    lspg_getcenter.dax) >= lsredis_getd( alignx
->precision)) ||
    (lspg_getcenter.day_isnull == 0 && fabs(
    lspg_getcenter.day) >= lsredis_getd( aligny
->precision)) ||
    (lspg_getcenter.daz_isnull == 0 && fabs(
    lspg_getcenter.daz) >= lsredis_getd( alignz
->precision))) {

lslogging_log_message( "md2cmds_rotate: requested
    positions cx %f, cy %f, ax %f, ay %f, az %f", cx, cy, ax, ay, az);

lslogging_log_message( "md2cmds_rotate: moving center"
    );
md2cmds_mvcenter_move( cx, cy, ax, ay, az);

lslogging_log_message( "md2cmds_rotate: waiting for
    center move");
lslogging_log_message( "md2cmds_rotate: done waiting")
    ;
}
lspg_getcenter_done();

if( md2cmds_home_wait( 20.0)) {
    lslogging_log_message( "md2cmds_rotate: homing motors
        timed out. Rotate aborted");
    lsevents_send_event( "Rotate Aborted");
    return;
}

if( md2cmds_move_wait( 20.0)) {
    lslogging_log_message( "md2cmds_rotate: moving motors
        timed out. Rotate aborted");
    lsevents_send_event( "Rotate Aborted");
    return;
}

// 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)", cx, cy, ax, ay, az, lspmac_getPosition
    (kappa), lspmac_getPosition( phi));

lslogging_log_message( "md2cmds_rotate: done with
    applycenter");
lspmac_video_rotate( 4.0);
lslogging_log_message( "md2cmds_rotate: starting
    rotation");
rotating = 1;
}

```

7.9.2.28 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 1148 of file md2cmds.c.

```

{
    struct tm t;
    int usecs;

    localtime_r( &(omega_zero_time.tv_sec), &t);

    usecs = omega_zero_time.tv_nsec / 1000;
    lspg_query_push( NULL, "SELECT px.trigcam('%d-%d-%d
        %d:%d:%d.%06d', %d, 0.0, 90.0)",
        t.tm_year+1900, t.tm_mon+1, t.tm_mday, t.tm_hour, t.tm_min,
        t.tm_sec, usecs,
        (int)(lspmac_getPosition( zoom)));
}

```

7.9.2.29 void md2cmds_run ()

Start up the thread.

Definition at line 1378 of file md2cmds.c.

```

{
    pthread_create( &md2cmds_thread, NULL,
        md2cmds_worker, NULL);
    lsevents_add_listener( "omega crossed zero",
        md2cmds_rotate_cb);
    lsevents_add_listener( "omega In Position",
        md2cmds_maybe_rotate_done_cb);
    lsevents_add_listener( ".+ (Moving|In Position)",
        md2cmds_maybe_done_moving_cb);
    lsevents_add_listener( "(.+) (Homing|Homed)",
        md2cmds_maybe_done_homing_cb);
    lsevents_add_listener( "capz (Moving|In Position)",
        md2cmds_time_capz_cb);
    lsevents_add_listener( "Coordsys 1 Stopped",
        md2cmds_coordsys_1_stopped_cb);
    lsevents_add_listener( "Coordsys 2 Stopped",
        md2cmds_coordsys_2_stopped_cb);
    lsevents_add_listener( "Coordsys 3 Stopped",
        md2cmds_coordsys_3_stopped_cb);
    lsevents_add_listener( "Coordsys 4 Stopped",
        md2cmds_coordsys_4_stopped_cb);
    lsevents_add_listener( "Coordsys 5 Stopped",
        md2cmds_coordsys_5_stopped_cb);
    lsevents_add_listener( "Coordsys 7 Stopped",
        md2cmds_coordsys_7_stopped_cb);
}

```

7.9.2.30 void md2cmds_set_scale_cb (char * event)

Fix up xscale and yscale when zoom changes.

Definition at line 1173 of file md2cmds.c.

```

{
    int mag;
    lsredis_obj_t *p1, *p2;
    char *vp;

    mag = lspmac_getPosition( zoom);

    p1 = lsredis_get_obj( "cam.xScale");
    p2 = lsredis_get_obj( "cam.zoom.%d.ScaleX", mag);

    vp = lsredis_getstr( p2);
}

```

```

lsredis_setstr( p2, vp);
free( vp);

p1 = lsredis_get_obj( "cam.yScale");
p2 = lsredis_get_obj( "cam.zoom.%d.ScaleY", mag);

vp = lsredis_getstr( p2);
lsredis_setstr( p2, vp);
free( vp);
}

```

7.9.2.31 void md2cmds_time_capz_cb (char * event)

Time the capillary motion for the transfer routine.

< track the time spent moving capz

Definition at line 1205 of file md2cmds.c.

```

{
static struct timespec capz_timestarted;
struct timespec now;
int nsec, sec;

if( strstr( event, "Moving") != NULL) {
clock_gettime( CLOCK_REALTIME, &capz_timestarted);
} else {
clock_gettime( CLOCK_REALTIME, &now);

sec = now.tv_sec - capz_timestarted.tv_sec;
nsec = 0;
if( now.tv_nsec > capz_timestarted.tv_nsec) {
sec--;
nsec += 1000000000;
}
nsec += now.tv_nsec - capz_timestarted.tv_nsec;
md2cmds_capz_moving_time = sec + nsec / 1000000000.
;
}
}

```

7.9.2.32 void md2cmds_transfer ()

Transfer a sample.

Definition at line 232 of file md2cmds.c.

```

{
int nextsample, abort_now;
double esttime;
double ax, ay, az, cx, cy, horz, vert, oref;
int err;

nextsample = lspg_nextsample_all( &err);
if( err) {
lslogging_log_message( "md2cmds_transfer: no sample
requested to be transfered, false alarm");
return;
}

//
// BLUMax sets up an abort dialogbox here. Probably we should figure out how
// we are going to handle that.
//

//
// Wait for motors to stop
//
if( md2cmds_is_moving()) {
lslogging_log_message( "md2cmds_transfer: Waiting for
previous motion to finish");
if( md2cmds_move_wait( 30.0)) {
lslogging_log_message( "md2cmds_transfer: Timed out
waiting for previous motion to finish. Aborting transfer");
}
}
}

```

```

//
// get positions we'll be needed to report to postgres
//
ax = lspmac_getPosition(algnx);
ay = lspmac_getPosition(algny);
az = lspmac_getPosition(algnz);
cx = lspmac_getPosition(cenx);
cy = lspmac_getPosition(ceny);
oref = lsredis_getd(lsredis_get_obj( "
    omega.reference")) * M_PI/180.;

horz = cx * cos(oref) + cy * sin(oref);
vert = cx * sin(oref) - cy * cos(oref);

if( lsredis_getd( capz->u2c) <= 0.0 || lsredis_getd
    ( capz->max_speed) <= 0.0 || lsredis_getd( capz->
    max_accel) <= 0.0) {
    esttime = 0.0;
} else {

    // Here we assume moving the capillary is the rate limiting step in
    // preparing the MD2.
    //
    // TODO: look at factors in which something besides the capillary determines
    // the time such as if the scintillator is out.
    //
    // pretend we are going to zero instead of the "Out" position. We should
    // probably arrange for
    // neutralPosition such that "Out" is zero.
    //
    // This also treats S curve acceleration as taking the same time as linear
    // acceleration.
    //
    esttime = lspmac_getPosition( capz)/lsredis_getd
        ( capz->u2c)/(lsredis_getd( capz->max_speed));
    // Time if we moved at constant velocity
    esttime += lsredis_getd( capz->max_speed)/
        lsredis_getd(capz->max_accel);
    // Correction for time spent accelerating
    esttime /= 1000.;
    // convert from milliseconds to seconds
}

lspg_starttransfer_call( nextsample,
    lspmac_getBIPosition( sample_detected), ax,
    ay, az, horz, vert, esttime);

// put the light down if it's not already
//
if( lspmac_getBIPosition( blight_down) != 1)
    blight_ud->moveAbs( blight_ud, 0);

// Pull the fluorescence detector out of the way
//
if( lspmac_getBIPosition( fluor_back) != 1)
    blight_ud->moveAbs( fluo, 0);

//
// Prepare for moving stuff
//
md2cmds_move_prep();

//
// Put the organs into position
//
md2cmds_organs_move_presets( "In", "Cover", "In",
    "Cover", "Cover");

md2cmds_home_prep();

//
// Home Kappa
//
lspmac_home1_queue( kappa);

//
// Home omega
//
lspmac_home1_queue( omega);

//
// wait for kappa cause we can't home phi until kappa's done
//
lspmac_moveabs_wait( kappa, 60.0);

```



```

//
// Home phi (whatever that means)
//
lspmac_home1_queue( phi);

// Now let's get back to postgresql (remember our query so long ago?)
//
lspg_starttransfer_wait();

//
// It's possible that the sample that's mounted is unknown to the robot.
// If so then we need to abort after we're done moving stuff
//
if( lspg_starttransfer.no_rows_returned ||
    lspg_starttransfer.starttransfer != 1)
    abort_now = 1;
else
    abort_now = 0;

lspg_starttransfer_done();

//
// Wait for the homing routines to finish
//
if( md2cmds_home_wait( 30.0)) {
    lslogging_log_message( "md2cmds_transfer: homing
        routines taking too long. Aborting transfer.");
    lsevents_send_event( "Transfer Aborted");
    return;
}

//
// Wait for all the motors to stop moving
//
if( md2cmds_move_wait( 30.0)) {
    lslogging_log_message( "md2cmds_transfer: We got bored
        waiting for the motors to stop. Aborting transfer. Later.");
    lsevents_send_event( "Transfer Aborted");
    return;
}

// TODO: check that all the motors are where we told them to go
//

if( abort_now) {
    lslogging_log_message( "md2cmds_transfer: Apparently
        there is a sample mounted already but we don't know where it is supposed to go");
    lsevents_send_event( "Transfer Aborted");
    return;
}

// refuse to go on if we do not have positive confirmation that the backlight
// is down and the
// fluorescence detector is back
//
if( lspmac_getBIPosition( blight_down) != 1 ||
    lspmac_getBIPosition( fluor_back) != 1) {
    lslogging_log_message( "md2cmds_transfer: It looks
        like either the back light is not down or the fluoescence dectector is not back");
    lsevents_send_event( "Transfer Aborted");
    return;
}

//
// Wait for the robot to unlock the cryo which signals us that we need to
// move the cryo back and drop air rights
//
lspg_waitcryo_all();

// Move the cryo back
//
cryo->moveAbs( cryo, 1);
lspmac_moveabs_wait( cryo, 10.0);

// simplest query yet!
lspg_query_push( lspg_waitcryo_cb, "SELECT
    px.dropairrights()");

// wait for the result
// TODO: find an easy way out of this in case of error
//
lspg_getcurrentsampleid_wait_for_id(
    nextsample);

// grab the airrights again
//

```

```

lspg_demandairrights_all();
lsevents_send_event( "Transfer Done");
}

```

7.9.2.33 void* md2cmds_worker (void * dummy)

Our worker thread.

Parameters

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

Definition at line 1307 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, "abort") == 0) {
md2cmds_abort();
} else if( strcmp( md2cmds_cmd, "transfer") == 0) {
md2cmds_transfer();
} else if( strcmp( md2cmds_cmd, "collect") == 0) {
md2cmds_collect();
} else if( strcmp( md2cmds_cmd, "rotate") == 0) {
md2cmds_rotate();
} else if( strcmp( md2cmds_cmd, "center") == 0) {
md2cmds_center();
} else if( strncmp( md2cmds_cmd, "moveAbs", 7) == 0) {
md2cmds_moveAbs( md2cmds_cmd);
} else if( strncmp( md2cmds_cmd, "changeMode", 10) == 0) {
md2cmds_phase_change( md2cmds_cmd);
}

md2cmds_cmd[0] = 0;
}
}

```

7.9.3 Variable Documentation

7.9.3.1 double md2cmds_capz_moving_time = NAN [static]

Definition at line 32 of file md2cmds.c.

7.9.3.2 char md2cmds_cmd[MD2CMDS_CMD_LENGTH]

our command;

Definition at line 24 of file md2cmds.c.

7.9.3.3 pthread_cond_t md2cmds_cond

condition to signal when it's time to run an md2 command

Definition at line 10 of file md2cmds.c.

7.9.3.4 pthread_cond_t md2cmds_homing_cond

coordinate homing and homed

Definition at line 18 of file md2cmds.c.

7.9.3.5 int md2cmds_homing_count = 0

We've asked a motor to home.

Definition at line 17 of file md2cmds.c.

7.9.3.6 pthread_mutex_t md2cmds_homing_mutex

our mutex;

Definition at line 19 of file md2cmds.c.

7.9.3.7 lsredis_obj_t* md2cmds_md_status_code

Definition at line 26 of file md2cmds.c.

7.9.3.8 pthread_cond_t md2cmds_moving_cond

wait for command to have been dequeued and run

coordinate call and response

Definition at line 14 of file md2cmds.c.

7.9.3.9 int md2cmds_moving_count = 0

Definition at line 22 of file md2cmds.c.

7.9.3.10 pthread_mutex_t md2cmds_moving_mutex

message passing between md2cmds and pg

Definition at line 15 of file md2cmds.c.

7.9.3.11 int md2cmds_moving_queue_wait = 0

Definition at line 13 of file md2cmds.c.

7.9.3.12 pthread_mutex_t md2cmds_mutex

mutex for the condition

Definition at line 11 of file md2cmds.c.

7.9.3.13 pthread_t md2cmds_thread [static]

Definition at line 28 of file md2cmds.c.

7.9.3.14 `int rotating = 0` `[static]`

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

Definition at line 30 of file md2cmds.c.

7.10 `mk_pgpmac_redis.py` File Reference

Namespaces

- namespace `mk_pgpmac_redis`

Variables

- list `mk_pgpmac_redis.head` `sys.argv[1]`
- list `mk_pgpmac_redis.pref_ini` `sys.argv[2]`
- list `mk_pgpmac_redis.hard_ini` `sys.argv[3]`
- dictionary `mk_pgpmac_redis.motor_dict`
- dictionary `mk_pgpmac_redis.hard_ini_fields`
- list `mk_pgpmac_redis.motor_field_lists`
- list `mk_pgpmac_redis.bi_list` `["CryoSwitch"]`
- dictionary `mk_pgpmac_redis.motor_presets`
- list `mk_pgpmac_redis.zoom_settings`
- tuple `mk_pgpmac_redis.hi` `iniParser.iniParser(hard_ini)`
- list `mk_pgpmac_redis.v` `motor_dict[m]`
- string `mk_pgpmac_redis.f` `"HSETNX"`
- tuple `mk_pgpmac_redis.pi` `iniParser.iniParser(pref_ini)`
- int `mk_pgpmac_redis.i` `0`
- tuple `mk_pgpmac_redis.ppos` `pi.get(section, option)`
- string `mk_pgpmac_redis.fnc` `"HSETNX"`
- tuple `mk_pgpmac_redis.b` `pi.get(section, "LightIntensity")`
- tuple `mk_pgpmac_redis.p` `pi.get(section, "MotorPosition")`
- tuple `mk_pgpmac_redis.x` `pi.get(section, "ScaleX")`
- tuple `mk_pgpmac_redis.y` `pi.get(section, "ScaleY")`

7.11 `pgpmac.c` File Reference

Main for the pgpmac project.

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

Functions

- void `stdinService` (struct pollfd *evt)
Handle keyboard input.
- void `pgpmac_printf` (char *fmt,...)
Terminal output routine ala printf.
- int `main` (int argc, char **argv)
Our main routine.

Variables

- WINDOW * [term_output](#)
place to print stuff out
- WINDOW * [term_input](#)
place to put the cursor
- WINDOW * [term_status](#)
shutter, lamp, air, etc status
- WINDOW * [term_status2](#)
shutter, lamp, air, etc status
- pthread_mutex_t [ncurses_mutex](#)
allow more than one thread access to the screen
- static struct pollfd [stdinfda](#)
Handle input from the keyboard.
- static int [running](#) = 1

7.11.1 Detailed Description

Main for the pgpmac project.

Date

2012

Author

Keith Brister

Copyright

All Rights Reserved

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

7.11.2 Function Documentation

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

Our main routine.

Parameters

in	<i>argc</i>	Number of arguments
in	<i>argv</i>	Vector of argument strings

Definition at line 353 of file pgpmac.c.

```

{
    static struct pollfd fda[3];           // input for poll: room for postgres,
        pmac, and stdin
    static int nfd = 0;                   // number of items in fda
    static int pollrtn = 0;
    static struct option long_options[] = {
        { "i-vars", 0, NULL, 'i' },
        { "m-vars", 0, NULL, 'm' },
        { NULL, 0, NULL, 0 }
    };
};

```

```

int c;
int ivars, mvars;
mvars=0;
ivars=0;

int i;                                // standard loop counter

while( 1) {
    c=getopt_long( argc, argv, "im", long_options, NULL);
    if( c == -1)
        break;

    switch( c) {
        case 'i':
            ivars=1;
            break;

        case 'm':
            mvars=1;
            break;

    }
}

stdinfd.fd = 0;
stdinfd.events = POLLIN;

initscr();                            // Start ncurses
raw();                                // Line buffering disabled, control
    chars trapped
keypad( stdscr, TRUE);                 // Why is F1 nifty?
refresh();

pthread_mutex_init( &ncurses_mutex, NULL);    // don't lock
    this mutex yet because we are not multi-threaded until the "_run" functions

//
// Since the modules reference objects in other modules it is important
// that everyone is initialized before anyone runs
//
lslogging_init();
lslogging_run();
lsevents_init();
lsevents_run();
lstimer_init();
lstimer_run();
lsredis_init( "MD2-21-ID-E", "redis\\.kvseq|stns\\.2\\.\\. (\\+)", "
    stns.2");
lsredis_run();

lspmac_init( ivars, mvars);
lspg_init();
md2cmds_init();

term_status = newwin( LS_DISPLAY_WINDOW_HEIGHT
    , LS_DISPLAY_WINDOW_WIDTH, 3*LS_DISPLAY_WINDOW_HEIGHT
    , 0*LS_DISPLAY_WINDOW_WIDTH);
box( term_status, 0, 0);
wnoutrefresh( term_status);

term_status2 = newwin( LS_DISPLAY_WINDOW_HEIGHT
    , LS_DISPLAY_WINDOW_WIDTH, 3*LS_DISPLAY_WINDOW_HEIGHT
    , 1*LS_DISPLAY_WINDOW_WIDTH);
box( term_status2, 0, 0);
wnoutrefresh( term_status2);

term_output = newwin( 20, 5*LS_DISPLAY_WINDOW_WIDTH
    , 4*LS_DISPLAY_WINDOW_HEIGHT, 0);
scrollok( term_output, 1);
wnoutrefresh( term_output);

term_input = newwin( 3, 5*LS_DISPLAY_WINDOW_WIDTH
    , 20+4*LS_DISPLAY_WINDOW_HEIGHT, 0);
box( term_input, 0, 0);
mvwprintw( term_input, 1, 1, "PMAC> ");
nodelay( term_input, TRUE);
keypad( term_input, TRUE);
wnoutrefresh( term_input);

doupdate();

lspmac_run();
lspg_run();
md2cmds_run();

while( running) {
    //

```

```

// Big loop
//

nfd = 0;

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

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

pollrtn = poll( fda, nfd, 10);

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

```

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

Terminal output routine ala printf.

Parameters

in	fmt	Printf style formatting string
----	-----	--------------------------------

Definition at line 330 of file pgpmac.c.

```

{
    va_list arg_ptr;

    pthread_mutex_lock( &ncurses_mutex);

    va_start( arg_ptr, fmt);
    vwprintw( term_output, fmt, arg_ptr);
    va_end( arg_ptr);

    wnoutrefresh( term_output);
    wnoutrefresh( term_input);
    doupdate();

    pthread_mutex_unlock( &ncurses_mutex);
}

```

7.11.2.3 void stdinService (struct pollfd *evt)

Handle keyboard input.

Parameters

in	evt	The pollfd object that caused this call
----	-----	---

Definition at line 254 of file pgpmac.c.

```

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

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

    switch( ch) {
case KEY_F(1):
case KEY_F(2):
case KEY_F(3):
    running = 0;
    break;

case 0x0001:      // Control-A
case 0x0002:      // Control-B
case 0x0003:      // Control-C
case 0x0004:      // Control-D
case 0x0005:      // Control-E
case 0x0006:      // Control-F
case 0x0007:      // Control-G
case 0x000b:      // Control-K
case 0x000f:      // Control-O
case 0x0010:      // Control-P
case 0x0011:      // Control-Q
case 0x0012:      // Control-R
case 0x0013:      // Control-Q
case 0x0016:      // Control-V
    cntrlcmd[0] = ch;
    cntrlcmd[1] = 0;
    lspmac_SockSendline( NULL, cntrlcmd);
    // PmacSockSendControlCharPrint( ch);
    break;

case KEY_BACKSPACE:
    cmds[cmds_on] = 0;
    cmds_on == 0 ? 0 : cmds_on--;
    break;

case KEY_ENTER:
case 0x000a:
    if( cmds_on > 0 && strlen( cmds) > 0) {
        lspmac_SockSendline( NULL, cmds);
    }
    memset( cmds, 0, sizeof( cmds));
    cmds_on = 0;
    break;

default:
    if( ch >= 0x20 && ch <= 0x7e) {
        if( cmds_on < sizeof( cmds)-1) {
            cmds[cmds_on++] = ch;
            cmds[cmds_on] = 0;
        }
    }
    break;
}

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

```

7.11.3 Variable Documentation

7.11.3.1 pthread_mutex_t ncurses_mutex

allow more than one thread access to the screen

Definition at line 242 of file pgpmac.c.

7.11.3.2 `int running = 1` `[static]`

Definition at line 249 of file pgpmac.c.

7.11.3.3 `struct pollfd stdinfd` `[static]`

Handle input from the keyboard.

Definition at line 248 of file pgpmac.c.

7.11.3.4 `WINDOW* term_input`

place to put the cursor

Definition at line 238 of file pgpmac.c.

7.11.3.5 `WINDOW* term_output`

place to print stuff out

Definition at line 237 of file pgpmac.c.

7.11.3.6 `WINDOW* term_status`

shutter, lamp, air, etc status

Definition at line 239 of file pgpmac.c.

7.11.3.7 `WINDOW* term_status2`

shutter, lamp, air, etc status

Definition at line 240 of file pgpmac.c.

7.12 pgpmac.h File Reference

Headers for the entire pgpmac project.

```
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netdb.h>
#include <string.h>
#include <netinet/in.h>
#include <errno.h>
#include <poll.h>
#include <libpq-fe.h>
#include <ncurses.h>
#include <math.h>
#include <pthread.h>
#include <signal.h>
#include <sys/signalfd.h>
#include <sys/time.h>
#include <time.h>
#include <getopt.h>
#include <regex.h>
#include <hiredis/hiredis.h>
#include <hiredis/async.h>
#include <search.h>
```

Data Structures

- struct [lsredis_obj_struct](#)
Redis Object Basic object whose value is synchronized with our redis db.
- struct [tagEthernetCmd](#)
PMAC ethernet packet definition.
- struct [lspmac_cmd_queue_struct](#)
PMAC command queue item.
- struct [lspmac_motor_struct](#)
Motor information.
- struct [lspmac_bi_struct](#)
Storage for binary inputs.
- struct [lspgQueryQueueStruct](#)
Store each query along with it's callback function.
- struct [lspg_waitcryo_struct](#)
- struct [lspg_getcurrentsampleid_struct](#)
- struct [lspg_demandairrights_struct](#)
- struct [lspg_getcenter_struct](#)
Storage for getcenter query Used for the md2 ROTATE command that generates the centering movies.
- struct [lspg_starttransfer_struct](#)
returns 1 if transfer can continue 0 to abort
- struct [lspg_nextsample_struct](#)
Returns the next sample number Just a 32 bit int (Ha!, take that, nextshot!)
- struct [lspg_nextshot_struct](#)
Storage definition for nextshot query.

Macros

- `#define _GNU_SOURCE`
- `#define LS_DISPLAY_WINDOW_HEIGHT 8`
Number of status box rows.
- `#define LS_DISPLAY_WINDOW_WIDTH 24`
Number of status box columns.
- `#define LS_PG_QUERY_STRING_LENGTH 1024`
Fixed length postgresql query strings. Queries should all be function calls so this is not as weird as one might think.
- `#define LSEVENTS_EVENT_LENGTH 256`
Fixed length for event names: simplifies string handling.
- `#define MD2CMDS_CMD_LENGTH 32`

Typedefs

- `typedef struct lsredis_obj_struct lsredis_obj_t`
Redis Object Basic object whose value is synchronized with our redis db.
- `typedef struct tagEthernetCmd pmac_cmd_t`
PMAC ethernet packet definition.
- `typedef struct lspmac_cmd_queue_struct pmac_cmd_queue_t`
PMAC command queue item.
- `typedef struct lspmac_motor_struct lspmac_motor_t`
Motor information.
- `typedef struct lspmac_bi_struct lspmac_bi_t`
Storage for binary inputs.
- `typedef struct lspgQueryQueueStruct lspg_query_queue_t`
Store each query along with it's callback function.
- `typedef struct lspg_waitcryo_struct lspg_waitcryo_t`
- `typedef struct lspg_getcurrentsampleid_struct lspg_getcurrentsampleid_t`
- `typedef struct lspg_demandairrights_struct lspg_demandairrights_t`
- `typedef struct lspg_getcenter_struct lspg_getcenter_t`
Storage for getcenter query Used for the md2 ROTATE command that generates the centering movies.
- `typedef struct lspg_starttransfer_struct lspg_starttransfer_t`
returns 1 if transfer can continue 0 to abort
- `typedef struct lspg_nextsample_struct lspg_nextsample_t`
Returns the next sample number Just a 32 bit int (Ha!, take that, nextshot!)
- `typedef struct lspg_nextshot_struct lspg_nextshot_t`
Storage definition for nextshot query.

Functions

- `double lspmac_getPosition (lspmac_motor_t *)`
get the motor position (with locking)
- `char ** lspg_array2ptrs (char *)`
returns a null terminated list of strings parsed from postgresql array
- `char ** lsredis_get_string_array (lsredis_obj_t *p)`

- void [lspmac_SockSendDPLine](#) (char *, char *fmt,...)
prepare (queue up) a line to send the dpram ascii command interface
- [pmac_cmd_queue_t * lspmac_SockSendline](#) (char *, char *,...)
Send a one line command.
- [lsredis_obj_t * lsredis_get_obj](#) (char *,...)
- char * [lsredis_getstr](#) (lsredis_obj_t *p)
return a copy of the key's string value
- void [PmacSockSendline](#) (char *s)
- unsigned int [lspg_nextsample_all](#) (int *err)
- long int [lsredis_getl](#) (lsredis_obj_t *p)
- void [lsevents_add_listener](#) (char *, void(*cb)(char *))
Add a callback routine to listen for a specific event.
- void [lsevents_init](#) ()
Initialize this module.
- void [lsevents_remove_listener](#) (char *, void(*cb)(char *))
Remove a listener previously added with lsevents_add_listener.
- void [lsevents_run](#) ()
Start up the thread and get out of the way.
- void [lsevents_send_event](#) (char *,...)
Call the callback routines for the given event.
- void [lslogging_init](#) ()
Initialize the lslogging objects.
- void [lslogging_log_message](#) (char *fmt,...)
The routine everyone will be talking about.
- void [lslogging_run](#) ()
Start up the worker thread.
- void [lspg_demandairrights_all](#) ()
do nothing until we get airrights
- void [lspg_getcenter_call](#) ()
Request a getcenter query.
- void [lspg_getcenter_done](#) ()
Done with getcenter query.
- void [lspg_getcenter_wait](#) ()
Wait for a getcenter query to return.
- void [lspg_getcurrentsampleid_wait_for_id](#) (unsigned int test)
- void [lspg_init](#) ()
Initiallize the lspg module.
- void [lspg_nextshot_call](#) ()
Queue up a nextshot query.
- void [lspg_nextshot_done](#) ()
Called when the next shot query has been processed.
- void [lspg_nextshot_wait](#) ()
Wait for the next shot query to get processed.
- void [lspg_query_push](#) (void(*cb)(lspg_query_queue_t *, PGresult *), char *fmt,...)
Place a query on the queue.
- void [lspg_run](#) ()
Start 'er runnin'.
- void [lspg_seq_run_prep_all](#) (long long skey, double [kappa](#), double [phi](#), double cx, double cy, double ax, double ay, double az)
Convinence function to call seq run prep.

- void [lspg_starttransfer_call](#) (unsigned int nextsample, int [sample_detected](#), double ax, double ay, double az, double horz, double vert, double esttime)
- void [lspg_starttransfer_done](#) ()
- void [lspg_starttransfer_wait](#) ()
- void [lspg_waitcryo_all](#) ()
 - no need to get fancy with the wait cryo command It should not return until the robot is almost ready for air rights*
- void [lspg_waitcryo_cb](#) ([lspg_query_queue_t](#) *qqp, PGresult *pgr)
- void [lspg_zoom_lut_call](#) ()
- int [lspmac_getBIPosition](#) ([lspmac_bi_t](#) *)
 - get binary input value*
- void [lspmac_home1_queue](#) ([lspmac_motor_t](#) *mp)
 - Home the motor.*
- void [lspmac_abort](#) ()
 - abort motion and try to recover*
- void [lspmac_init](#) (int, int)
 - Initialize this module.*
- int [lspmac_jogabs_queue](#) ([lspmac_motor_t](#) *, double)
 - Use jog to move motor to requested position.*
- int [lspmac_move_or_jog_abs_queue](#) ([lspmac_motor_t](#) *mp, double requested_position, int use_jo)
 - Move method for normal stepper and servo motor objects Returns non-zero on abort, zero if OK.*
- int [lspmac_move_or_jog_preset_queue](#) ([lspmac_motor_t](#) *, char *, int)
 - move using a preset value returns 0 on success, non-zero on error*
- void [lspmac_move_or_jog_queue](#) ([lspmac_motor_t](#) *, double, int)
- int [lspmac_move_preset_queue](#) ([lspmac_motor_t](#) *mp, char *preset_name)
 - Move a given motor to one of its preset positions.*
- int [lspmac_moveabs_queue](#) ([lspmac_motor_t](#) *, double)
 - Use coordinate system motion program, if available, to move motor to requested position.*
- int [lspmac_moveabs_wait](#) ([lspmac_motor_t](#) *mp, double timeout)
 - Wait for motor to finish moving.*
- void [lspmac_run](#) ()
 - Start up the lspmac thread.*
- void [lspmac_video_rotate](#) (double secs)
 - Special motion program to collect centering video.*
- int [lsredis_cmpnstr](#) ([lsredis_obj_t](#) *p, char *s, int n)
- int [lsredis_cmpstr](#) ([lsredis_obj_t](#) *p, char *s)
- int [lsredis_find_preset](#) (char *base, char *preset_name, double *dval)
- int [lsredis_getb](#) ([lsredis_obj_t](#) *p)
- double [lsredis_getd](#) ([lsredis_obj_t](#) *p)
- void [lsredis_init](#) (char *pub, char *re, char *head)
 - Initialize this module, that is, set up the connections.*
- int [lsredis_regexec](#) (const regex_t *preg, [lsredis_obj_t](#) *p, size_t nmatch, regmatch_t *pmatch, int eflags)
- void [lsredis_run](#) ()
- void [lsredis_setstr](#) ([lsredis_obj_t](#) *p, char *fmt,...)
 - Set the value and update redis.*
- void [lstimer_add_timer](#) (char *, int, unsigned long int, unsigned long int)
 - Create a timer.*
- void [lstimer_init](#) ()
 - Initialize the timer list and pthread stuff.*
- void [lstimer_run](#) ()
 - Start up our thread.*
- void [lsupdate_init](#) ()
- void [lsupdate_run](#) ()

- void `md2cmds_init` ()
Initialize the md2cmds module.
- void `md2cmds_run` ()
Start up the thread.
- void `pgpmac_printf` (char *fmt,...)
Terminal output routine ala printf.

Variables

- `lspg_waitcryo_t` `lspg_waitcryo`
signal the robot
- `lspg_getcurrentsampleid_t` `lspg_getcurrentsampleid`
our currentsample id
- `lspg_demandairrights_t` `lspg_demandairrights`
our demandairrights object
- `lspg_getcenter_t` `lspg_getcenter`
the getcenter object
- `lspg_starttransfer_t` `lspg_starttransfer`
start a sample transfer
- `lspg_nextsample_t` `lspg_nextsample`
the very next sample
- `lspg_nextshot_t` `lspg_nextshot`
the nextshot object
- `lspmac_motor_t` `lspmac_motors` []
All our motors.
- `lspmac_motor_t` * `omega`
MD2 omega axis (the air bearing)
- `lspmac_motor_t` * `alignx`
Alignment stage X.
- `lspmac_motor_t` * `aligny`
Alignment stage Y.
- `lspmac_motor_t` * `alignz`
Alignment stage X.
- `lspmac_motor_t` * `anal`
Polaroid analyzer motor.
- `lspmac_motor_t` * `zoom`
Optical zoom.
- `lspmac_motor_t` * `apery`
Aperture Y.
- `lspmac_motor_t` * `aperz`
Aperture Z.
- `lspmac_motor_t` * `capy`
Capillary Y.
- `lspmac_motor_t` * `capz`
Capillary Z.
- `lspmac_motor_t` * `scint`
Scintillator Z.
- `lspmac_motor_t` * `cenx`
Centering Table X.
- `lspmac_motor_t` * `ceny`

- Centering Table Y.
- `lspmac_motor_t * kappa`
- Kappa.
- `lspmac_motor_t * phi`
- Phi (not data collection axis)
- `lspmac_motor_t * fshut`
- Fast shutter.
- `lspmac_motor_t * flight`
- Front Light DAC.
- `lspmac_motor_t * blight`
- Back Light DAC.
- `lspmac_motor_t * fscint`
- Scintillator Piezo DAC.
- `lspmac_motor_t * smart_mag_oo`
- Smart Magnet on/off.
- `lspmac_motor_t * blight_ud`
- Back light Up/Down actuator.
- `lspmac_motor_t * cryo`
- Move the cryostream towards or away from the crystal.
- `lspmac_motor_t * dryer`
- blow air on the scintillator to dry it off
- `lspmac_motor_t * fluo`
- Move the fluorescence detector in/out.
- `lspmac_motor_t * flight_oo`
- Turn front light on/off.
- `lspmac_motor_t * blight_f`
- Back light scale factor.
- `lspmac_motor_t * flight_f`
- Front light scale factor.
- `int lspmac_nmotors`
- The number of motors we manage.
- `lspmac_bi_t * lp_air`
- Low pressure air OK.
- `lspmac_bi_t * hp_air`
- High pressure air OK.
- `lspmac_bi_t * cryo_switch`
- that little toggle switch for the cryo
- `lspmac_bi_t * blight_down`
- Backlight is down.
- `lspmac_bi_t * blight_up`
- Backlight is up.
- `lspmac_bi_t * cryo_back`
- cryo is in the back position
- `lspmac_bi_t * fluor_back`
- fluor is in the back position
- `lspmac_bi_t * sample_detected`
- smart magnet detected sample
- `lspmac_bi_t * etel_ready`
- ETEL is ready.
- `lspmac_bi_t * etel_on`
- ETEL is on.

- [lspmac_bi_t * etel_init_ok](#)
ETEL initialized OK.
- [lspmac_bi_t * minikappa_ok](#)
Minikappa is OK (whatever that means)
- [lspmac_bi_t * smart_mag_on](#)
smart magnet is on
- [lspmac_bi_t * arm_parked](#)
(whose arm? parked where?)
- [lspmac_bi_t * shutter_open](#)
shutter is open (note in pmc says this is a slow input)
- [lspmac_bi_t * smart_mag_off](#)
smart magnet is off
- [lspmac_bi_t * smart_mag_err](#)
smart magnet error (coil broken perhaps)
- struct timespec [omega_zero_time](#)
Time we believe that omega crossed zero.
- WINDOW * [term_output](#)
place to print stuff out
- WINDOW * [term_input](#)
place to put the cursor
- WINDOW * [term_status](#)
shutter, lamp, air, etc status
- WINDOW * [term_status2](#)
shutter, lamp, air, etc status
- pthread_mutex_t [ncurses_mutex](#)
allow more than one thread access to the screen
- pthread_cond_t [md2cmds_cond](#)
condition to signal when it's time to run an md2 command
- pthread_mutex_t [md2cmds_mutex](#)
mutex for the condition
- pthread_cond_t [md2cmds_pg_cond](#)
- pthread_mutex_t [md2cmds_pg_mutex](#)
- pthread_mutex_t [pmac_queue_mutex](#)
manage access to the pmac command queue
- pthread_cond_t [pmac_queue_cond](#)
wait for a command to be sent to PMAC before continuing
- pthread_mutex_t [lspmac_shutter_mutex](#)
Coordinates threads reading shutter status.
- pthread_cond_t [lspmac_shutter_cond](#)
Allows waiting for the shutter status to change.
- int [lspmac_shutter_state](#)
State of the shutter, used to detect changes.
- int [lspmac_shutter_has_opened](#)
Indicates that the shutter had opened, perhaps briefly even if the state did not change.
- pthread_mutex_t [lspmac_moving_mutex](#)
Coordinate moving motors between threads.
- pthread_cond_t [lspmac_moving_cond](#)
Wait for motor(s) to finish moving condition.
- int [lspmac_moving_flags](#)
Flag used to implement motor moving condition.
- pthread_mutex_t [md2_status_mutex](#)

Synchronize reading/writing status buffer.

- char [md2cmds_cmd](#) []
our command;
- [lsredis_obj_t](#) * [md2cmds_md_status_code](#)

7.12.1 Detailed Description

Headers for the entire pgpmac project.

Date

2012

Author

Keith Brister

Copyright

All Rights Reserved

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

7.12.2 Macro Definition Documentation

7.12.2.1 #define _GNU_SOURCE

Definition at line 7 of file [pgpmac.h](#).

7.12.2.2 #define LS_DISPLAY_WINDOW_HEIGHT 8

Number of status box rows.

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

7.12.2.3 #define LS_DISPLAY_WINDOW_WIDTH 24

Number of status box columns.

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

7.12.2.4 #define LS_PG_QUERY_STRING_LENGTH 1024

Fixed length postgresql query strings. Queries should all be function calls so this is not as weird as one might think.

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

7.12.2.5 #define LSEVENTS_EVENT_LENGTH 256

Fixed length for event names: simplifies string handling.

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

7.12.2.6 `#define MD2CMDS_CMD_LENGTH 32`

Definition at line 484 of file pgpmac.h.

7.12.3 Typedef Documentation

7.12.3.1 `typedef struct lspg_demandairrights_struct lspg_demandairrights_t`

7.12.3.2 `typedef struct lspg_getcenter_struct lspg_getcenter_t`

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

7.12.3.3 `typedef struct lspg_getcurrentsampleid_struct lspg_getcurrentsampleid_t`

7.12.3.4 `typedef struct lspg_nextsample_struct lspg_nextsample_t`

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

7.12.3.5 `typedef struct lspg_nextshot_struct lspg_nextshot_t`

Storage definition for nextshot query.

The next shot query returns all the information needed to collect the next data frame. Since SQL allows for null fields independently from blank strings a separate integer is used as a flag for this case. This adds to the program complexity but allows for some important cases. Suck it up.definition of the next image to be taken (and the one after that, too!)

7.12.3.6 `typedef struct lspgQueryQueueStruct lspg_query_queue_t`

Store each query along with it's callback function.

All calls are asynchronous

7.12.3.7 `typedef struct lspg_starttransfer_struct lspg_starttransfer_t`

returns 1 if transfer can continue 0 to abort

7.12.3.8 `typedef struct lspg_waitcryo_struct lspg_waitcryo_t`

7.12.3.9 `typedef struct lspmac_bi_struct lspmac_bi_t`

Storage for binary inputs.

7.12.3.10 `typedef struct lspmac_motor_struct lspmac_motor_t`

Motor information.

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

7.12.3.11 `typedef struct lsredis_obj_struct lsredis_obj_t`

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

7.12.3.12 typedef struct lspmac_cmd_queue_struct pmac_cmd_queue_t

PMAC command queue item.

Command queue items are fixed length to simplify memory management.

7.12.3.13 typedef struct tagEthernetCmd pmac_cmd_t

PMAC ethernet packet definition.

Taken directly from the Delta Tau documentation.

7.12.4 Function Documentation

7.12.4.1 void lsevents_add_listener (char * event, void (*)(char *) cb)

Add a callback routine to listen for a specific event.

Parameters

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

Definition at line 75 of file lsevents.c.

```

{
    lsevents_listener_t *new;
    int err;
    char *errbuf;
    int nerrbuf;

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

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

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

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

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

```

7.12.4.2 void lsevents_init ()

Initialize this module.

Definition at line 214 of file lsevents.c.

```

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

```

7.12.4.3 void lsevents_remove_listener (char * event, void(*)(char *) cb)

Remove a listener previously added with lsevents_add_listener.

Parameters

<i>event</i>	The name of the event
<i>cb</i>	The callback routine to remove

Definition at line 120 of file lsevents.c.

```

    {
        lsevents_listener_t *last, *current;

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

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

```

7.12.4.4 void lsevents_run ()

Start up the thread and get out of the way.

Definition at line 222 of file lsevents.c.

```

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

```

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

Call the callback routines for the given event.

Parameters

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

Definition at line 45 of file lsevents.c.

```

{
    char event[LSEVENTS_EVENT_LENGTH];
    va_list arg_ptr;

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

    pthread_mutex_lock( &lsevents_queue_mutex);

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

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

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

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

```

7.12.4.6 void lslogging_init ()

Initialize the lslogging objects.

Definition at line 37 of file lslogging.c.

```

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

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

```

7.12.4.7 void lslogging_log_message (char * *fmt*, ...)

The routine everyone will be talking about.

Parameters

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

Definition at line 48 of file lslogging.c.

```

{
    char msg[LSLOGGING_MSG_LENGTH];

```

```

struct timespec theTime;
va_list arg_ptr;
unsigned int on;

clock_gettime( CLOCK_REALTIME, &theTime);

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

pthread_mutex_lock( &lslogging_mutex);

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

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

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

```

7.12.4.8 void lslogging_run ()

Start up the worker thread.

Definition at line 105 of file lslogging.c.

```

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

```

7.12.4.9 char** lspg_array2ptrs (char *)

returns a null terminated list of strings parsed from postgresql array

Definition at line 161 of file lspg.c.

```

{
char **rtn, *sp, *acums;
int i, n, inquote, havebackslash, rtni;;
int mxsz;

inquote = 0;
havebackslash = 0;

// Dispense with the null input condition before we complicate the code below
if( a == NULL || a[0] != '{' || a[strlen(a)-1] != '}')
return NULL;

// Count the maximum number of strings
// Actual number will be less if there are quoted commas
//
n = 1;
for( i=0; a[i]; i++) {
if( a[i] == ',' )
n++;
}
//
// The maximum size of any string is the length of a (+1)
//
mxsz = strlen(a) + 1;

// This is the accumulation string to make up the array elements
acums = (char *)calloc( mxsz, sizeof( char));
if( acums == NULL) {
lslogging_log_message( "lspg_array2ptrs: out of memory
(acums)");
}
}

```

```

    exit( 1);
}

//
// allocate storage for the pointer array and the null terminator
//
rtn = (char **)calloc( n+1, sizeof( char *));
if( rtn == NULL) {
    lslogging_log_message( "lspg_array2ptrs: out of memory
(rtn)");
    exit( 1);
}
rtni = 0;

// Go through and create the individual strings
sp = acums;
*sp = 0;

inquote = 0;
havebackslash = 0;
for( i=1; a[i] != 0; i++) {
    switch( a[i]) {
        case '"':
            if( havebackslash) {
                // a quoted quote. Cool
                //
                *(sp++) = a[i];
                *sp = 0;
                havebackslash = 0;
            } else {
                // Toggle the flag
                inquote = 1 - inquote;
            }
            break;

        case '\\':
            if( havebackslash) {
                *(sp++) = a[i];
                *sp = 0;
                havebackslash = 0;
            } else {
                havebackslash = 1;
            }
            break;

        case ',':
            if( inquote || havebackslash) {
                *(sp++) = a[i];
                *sp = 0;
                havebackslash = 0;
            } else {
                rtn[rtni++] = strdup( acums);
                sp = acums;
            }
            break;

        case '}':
            if( inquote || havebackslash) {
                *(sp++) = a[i];
                *sp = 0;
                havebackslash = 0;
            } else {
                rtn[rtni++] = strdup( acums);
                rtn[rtni] = NULL;
                free( acums);
                return( rtn);
            }
            break;

        default:
            *(sp++) = a[i];
            *sp = 0;
            havebackslash = 0;
    }
}
//
// Getting here means the final '}' was missing
// Probably we should throw an error or log it or something.
// Through out the last entry since this there is not resonable expectation
// that
// we should be parsing it anyway.
//
rtn[rtni] = NULL;
free( acums);
return( rtn);
}

```

7.12.4.10 void lspg_demandairrights_all ()

do nothing until we get airrights

Definition at line 556 of file lspg.c.

```

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

```

7.12.4.11 void lspg_getcenter_call ()

Request a getcenter query.

Definition at line 1177 of file lspg.c.

```

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

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

```

7.12.4.12 void lspg_getcenter_done ()

Done with getcenter query.

Definition at line 1195 of file lspg.c.

```

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

```

7.12.4.13 void lspg_getcenter_wait ()

Wait for a getcenter query to return.

Definition at line 1187 of file lspg.c.

```

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

```

7.12.4.14 void lspg_getcurrentsampleid_wait_for_id (unsigned int test)

Definition at line 393 of file lspg.c.

```

    {
        pthread_mutex_lock( &lspg_getcurrentsampleid.mutex
        );
        while( lspg_getcurrentsampleid.getcurrentsampleid
            != test)
            pthread_cond_wait( &lspg_getcurrentsampleid.cond
                , &lspg_getcurrentsampleid.mutex);

        pthread_mutex_unlock( &lspg_getcurrentsampleid.mutex
        );
    }

```


7.12.4.15 void lspg_init ()

Initiallize the lspg module.

Definition at line 1758 of file lspg.c.

```

{
pthread_mutex_init( &lspg_queue_mutex, NULL);
pthread_cond_init( &lspg_queue_cond, NULL);

lspg_demandairrights_init();
lspg_getcenter_init();
lspg_getcurrentsampleid_init();
lspg_lock_detector_init();
lspg_lock_diffractionmeter_init();
lspg_nextsample_init();
lspg_nextshot_init();
lspg_seq_run_prep_init();
lspg_starttransfer_init();
lspg_wait_for_detector_init();
lspg_waitcryo_init();
}

```

7.12.4.16 unsigned int lspg_nextsample_all (int * err)

Definition at line 468 of file lspg.c.

```

{
unsigned int rtn;

lspg_nextsample_call();
lspg_nextsample_wait();

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

return rtn;
}

```

7.12.4.17 void lspg_nextshot_call ()

Queue up a nextshot query.

Definition at line 824 of file lspg.c.

```

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

lspg_query_push( lspg_nextshot_cb, "SELECT *
FROM px.nextshot2()");
}

```

7.12.4.18 void lspg_nextshot_done ()

Called when the next shot query has been processed.

Definition at line 842 of file lspg.c.

```

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

```

7.12.4.19 void lspg_nextshot_wait ()

Wait for the next shot query to get processed.

Definition at line 834 of file lspg.c.

```

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

```

7.12.4.20 void lspg_query_push (void(*) (lspg_query_queue_t *, PGresult *) cb, char * fmt, ...)

Place a query on the queue.

Parameters

in	<i>cb</i>	Our callback function that deals with the response
in	<i>fmt</i>	Printf style function to generate the query

Definition at line 234 of file kvredis.c.

```

    {
        int idx;
        va_list arg_ptr;

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

        idx = lspg_query_queue_on % LS_PG_QUERY_QUEUE_LENGTH
            ;

        va_start( arg_ptr, fmt);
        vsnprintf( lspg_query_queue[idx].qs,
            LS_PG_QUERY_STRING_LENGTH-1, fmt, arg_ptr);
        va_end( arg_ptr);

        lspg_query_queue[idx].qs[LS_PG_QUERY_STRING_LENGTH
            - 1] = 0;
        lspg_query_queue[idx].onResponse = cb;
        lspg_query_queue_on++;
    }
};

```

7.12.4.21 void lspg_run ()

Start 'er runnin'.

Definition at line 1777 of file lspg.c.

```

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

```

```
    lspmac_sample_detector_cb);
}
```

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

Convenience function to call seq run prep.

Parameters

in	<i>skey</i>	px.shots key for this image
in	<i>kappa</i>	current kappa postion
in	<i>phi</i>	current phi postition
in	<i>cx</i>	current center table x
in	<i>cy</i>	current center table y
in	<i>ax</i>	current alignment table x
in	<i>ay</i>	current alignment table y
in	<i>az</i>	current alignment table z

Definition at line 1095 of file `lspg.c`.

```
    {
    lspg_seq_run_prep_call( skey, kappa, phi, cx,
        cy, ax, ay, az);
    lspg_seq_run_prep_wait();
    lspg_seq_run_prep_done();
    }
```

7.12.4.23 void `lspg_starttransfer_call` (unsigned int *nextsample*, int *sample_detected*, double *ax*, double *ay*, double *az*, double *horz*, double *vert*, double *esttime*)

Definition at line 302 of file `lspg.c`.

```

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

lspg_query_push( lspg_starttransfer_cb, "
    SELECT px.starttransfer( %d, %d, %.3f, %.3f, %.3f, %.3f, %.3f, %.3f",
        nextsample, sample_detected, ax, ay, az, horz
        , vert, esttime);
}
```

7.12.4.24 void `lspg_starttransfer.done` ()

Definition at line 317 of file `lspg.c`.

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

7.12.4.25 void `lspg_starttransfer.wait` ()

Definition at line 311 of file `lspg.c`.

```

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

```

7.12.4.26 void lspg_waitcryo_all ()

no need to get fancy with the wait cryo command It should not return until the robot is almost ready for air rights

Definition at line 507 of file lspg.c.

```

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

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

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

pthread_mutex_unlock( &lspg_waitcryo.mutex);
}

```

7.12.4.27 void lspg_waitcryo_cb (lspg_query_queue_t * qq, PGresult * pgr)

Definition at line 497 of file lspg.c.

```

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

```

7.12.4.28 void lspg_zoom_lut.call ()

7.12.4.29 void lspmac_abort ()

abort motion and try to recover

Definition at line 2012 of file lspmac.c.

```

    {
//
// Stop everything! (consider ^O instead of ^A)
//
lspmac_SockSendDPControlChar( "Abort Request", 0
x01);
}

```

7.12.4.30 int lspmac_getBIPosition (lspmac_bi_t *)

get binary input value

Definition at line 1579 of file lspmac.c.

```

    {
    int rtn;
    pthread_mutex_lock( &bip->mutex);
    rtn = bip->position;
    pthread_mutex_unlock( &bip->mutex);
    return rtn;
}

```

7.12.4.31 double lspmac_getPosition (lspmac_motor_t * mp)

get the motor position (with locking)

Parameters

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

Definition at line 1339 of file lspmac.c.

```

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

```

7.12.4.32 void lspmac_home1_queue (lspmac_motor_t * mp)

Home the motor.

Parameters

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

Definition at line 1205 of file lspmac.c.

```

    {
    int i;
    int motor_num;
    int coord_num;
    char **home;

    pthread_mutex_lock( &(mp->mutex));

    motor_num = lsredis_get1( mp->motor_num);
    coord_num = lsredis_get1( mp->coord_num);
    home      = lsredis_get_string_array( mp->home);

    // Each of the motors should have this defined
    // but let's not seg fault if home is missing
    //
    if( home == NULL || *home == NULL) {
        //
        // Note we are already initialized
        // so if we are here there is something wrong.
        //
        lslogging_log_message( "lspmac_home1_queue: null or
            empty home strings for motor %s", mp->name);
        pthread_mutex_unlock( &(mp->mutex));
        return;
    }

    // We've already been called. Don't home again until
    // we're finish with the last time.
    //
    if( mp->homing) {
        pthread_mutex_unlock( &(mp->mutex));
        return;
    }
}

```

```

//
// Don't go on if any other motors in this coordinate system are homing.
// It's possible to write the homing program to home all the motors in the
// coordinate
// system. TODO (hint hint)
//
if( coord_num > 0) {
    for( i=0; i<lspmac_nmotors; i++) {
        if( &(lspmac_motors[i]) == mp)
            continue;
        if( lsredis_getl(lspmac_motors[i].coord_num) ==
            coord_num) {
            int nogo;
            nogo = 0;
            pthread_mutex_lock( &(lspmac_motors[i].mutex));
            //
            // Don't go on if
            //
            // we are homing or ( not in position
            while in open loop)
            //
            if( lspmac_motors[i].homing || (((lspmac_motors
            [i].status2 & 0x01)==0) && ((lspmac_motors[i].status1 & 0x040000)
            != 0)))
                nogo = 1;
            pthread_mutex_unlock( &(lspmac_motors[i].mutex));
            if( nogo) {
                pthread_mutex_unlock( &(mp->mutex));
                return;
            }
        }
    }
}
mp->homing = 1;
mp->not_done = 1; // set up waiting for cond
mp->motion_seen = 0;
// This opens the control loop.
// The status routine should notice this and the fact that
// the homing flag is set and call on the home2 routine
//
// Only send the open loop command if we are not in
// open loop mode already. This test might prevent a race condition
// where we've already moved the home2 routine (and queue the homing program
// motion)
// before the open loop command is dequeued and acted on.
//
if( ~(mp->status1) & 0x040000) {
    lspmac_SockSendDPline( mp->name, "##d$*",
        motor_num);
}

pthread_mutex_unlock( &(mp->mutex));

lsevents_send_event( "%s Homing", mp->name);
}

```

7.12.4.33 void lspmac_init (int , int)

Initialize this module.

Definition at line 3132 of file lspmac.c.

```

{
    md2_status_t *p;
    pthread_mutexattr_t mutex_initializer;

    // Set our global harvest flags
    getivars = ivarsflag;
    getmvars = mvarsflag;

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

    // All important status mutex
    pthread_mutex_init( &md2_status_mutex, &mutex_initializer);

    //
    // Get the MD2 initialization strings
    //
    // lspmac_md2_init = lsredis_get_obj( "md2_pmac.init"); // hard coded now.
}

```

```

//
// Initialize the motor objects
//

p = &md2_status;

omega = lspmac_motor_init( &(lspmac_motors
[ 0]), 0, 0, &p->omega_act_pos, &p->omega_status_1
, &p->omega_status_2, "Omega #1 &1 X", "omega",
lspmac_moveabs_queue);
alignx = lspmac_motor_init( &(lspmac_motors
[ 1]), 0, 1, &p->alignx_act_pos, &p->alignx_status_1
, &p->alignx_status_2, "Align X #2 &3 X", "align.x",
lspmac_moveabs_queue);
aligny = lspmac_motor_init( &(lspmac_motors
[ 2]), 0, 2, &p->aligny_act_pos, &p->aligny_status_1
, &p->aligny_status_2, "Align Y #3 &3 Y", "align.y",
lspmac_moveabs_queue);
alignz = lspmac_motor_init( &(lspmac_motors
[ 3]), 0, 3, &p->alignz_act_pos, &p->alignz_status_1
, &p->alignz_status_2, "Align Z #4 &3 Z", "align.z",
lspmac_moveabs_queue);
anal = lspmac_motor_init( &(lspmac_motors
[ 4]), 0, 4, &p->analyzer_act_pos, &p->analyzer_status_1
, &p->analyzer_status_2, "Anal #5", "lightPolar",
lspmac_moveabs_queue);
zoom = lspmac_motor_init( &(lspmac_motors
[ 5]), 1, 0, &p->zoom_act_pos, &p->zoom_status_1
, &p->zoom_status_2, "Zoom #6 &4 Z", "cam.zoom",
lspmac_movezoom_queue);
apery = lspmac_motor_init( &(lspmac_motors
[ 6]), 1, 1, &p->aperturey_act_pos, &p->aperturey_status_1
, &p->aperturey_status_2, "Aper Y #7 &5 Y", "appy",
lspmac_moveabs_queue);
aperz = lspmac_motor_init( &(lspmac_motors
[ 7]), 1, 2, &p->aperturez_act_pos, &p->aperturez_status_1
, &p->aperturez_status_2, "Aper Z #8 &5 Z", "appz",
lspmac_moveabs_queue);
capy = lspmac_motor_init( &(lspmac_motors
[ 8]), 1, 3, &p->capy_act_pos, &p->capy_status_1
, &p->capy_status_2, "Cap Y #9 &5 U", "capy",
lspmac_moveabs_queue);
capz = lspmac_motor_init( &(lspmac_motors
[ 9]), 1, 4, &p->capz_act_pos, &p->capz_status_1
, &p->capz_status_2, "Cap Z #10 &5 V", "capz",
lspmac_moveabs_queue);
scint = lspmac_motor_init( &(lspmac_motors
[10]), 2, 0, &p->scint_act_pos, &p->scint_status_1
, &p->scint_status_2, "Scin Z #11 &5 W", "scint",
lspmac_moveabs_queue);
cenx = lspmac_motor_init( &(lspmac_motors
[11]), 2, 1, &p->centerx_act_pos, &p->centerx_status_1
, &p->centerx_status_2, "Cen X #17 &2 X", "centering.x",
lspmac_moveabs_queue);
ceny = lspmac_motor_init( &(lspmac_motors
[12]), 2, 2, &p->centery_act_pos, &p->centery_status_1
, &p->centery_status_2, "Cen Y #18 &2 Y", "centering.y",
lspmac_moveabs_queue);
kappa = lspmac_motor_init( &(lspmac_motors
[13]), 2, 3, &p->kappa_act_pos, &p->kappa_status_1
, &p->kappa_status_2, "Kappa #19 &7 X", "kappa",
lspmac_moveabs_queue);
phi = lspmac_motor_init( &(lspmac_motors[
14]), 2, 4, &p->phi_act_pos, &p->phi_status_1
, &p->phi_status_2, "Phi #20 &7 Y", "phi",
lspmac_moveabs_queue);

fshut = lspmac_fshut_init( &(lspmac_motors
[15]));
flight = lspmac_dac_init( &(lspmac_motors[1
6]), &p->front_dac, "M1200", "frontLight.intensity",
lspmac_movedac_queue);
blight = lspmac_dac_init( &(lspmac_motors[1
7]), &p->back_dac, "M1201", "backLight.intensity",
lspmac_movedac_queue);
fscint = lspmac_dac_init( &(lspmac_motors[1
8]), &p->scint_piezo, "M1203", "scint.focus",
lspmac_movedac_queue);

smart_mag_oo = lspmac_bo_init( &(lspmac_motors
[19]), "smartMagnet", "M1100=%d", &(md2_status.acc11c_5), 0x01
);
blight_ud = lspmac_bo_init( &(lspmac_motors
[20]), "backLight", "M1101=%d", &(md2_status.acc11c_5), 0x02
);
cryo = lspmac_bo_init( &(lspmac_motors

```

```

    [21]), "cryo",          "M1102=%d", &(md2_status.acc11c_5), 0x04)
;
dryer      = lspmac_bo_init( &(lspmac_motors
[22]), "dryer",          "M1103=%d", &(md2_status.acc11c_5), 0x08)
;
fluor      = lspmac_bo_init( &(lspmac_motors
[23]), "fluor",          "M1104=%d", &(md2_status.acc11c_5), 0x10)
;
flight_oo  = lspmac_soft_motor_init( &(
    lspmac_motors[24]), "frontLight",
    lspmac_moveabs_frontlight_oo_queue);
blight_f   = lspmac_soft_motor_init( &(
    lspmac_motors[25]), "backLight.factor",
    lspmac_moveabs_blight_factor_queue);
flight_f    = lspmac_soft_motor_init( &(
    lspmac_motors[26]), "frontLight.factor",
    lspmac_moveabs_flight_factor_queue);

lp_air      = lspmac_bi_init( &(lspmac_bis[
    0]), &(md2_status.acc11c_1), 0x01, "Low Pressure Air OK", "
    Low Pressure Air Failed");
hp_air      = lspmac_bi_init( &(lspmac_bis[
    1]), &(md2_status.acc11c_1), 0x02, "High Pressure Air OK", "
    High Pressure Air Failed");
cryo_switch = lspmac_bi_init( &(lspmac_bis
[ 2]), &(md2_status.acc11c_1), 0x04, "CryoSwitchChanged",
    "CryoSwitchChanged");
blight_down = lspmac_bi_init( &(lspmac_bis
[ 3]), &(md2_status.acc11c_1), 0x08, "Backlight Down",
    "Backlight Not Down");
blight_up   = lspmac_bi_init( &(lspmac_bis
[ 4]), &(md2_status.acc11c_1), 0x10, "Backlight Up",
    "Backlight Not Up");
cryo_back    = lspmac_bi_init( &(lspmac_bis
[ 5]), &(md2_status.acc11c_1), 0x40, "Cryo Back",
    "Cryo Not Back");
fluor_back   = lspmac_bi_init( &(lspmac_bis
[ 6]), &(md2_status.acc11c_1), 0x01, "Fluor. Det. Parked",
    "Fluor. Det. Not Parked");
sample_detected = lspmac_bi_init( &(lspmac_bis
[ 7]), &(md2_status.acc11c_2), 0x02, "SamplePresent",
    "SampleAbsent");
etel_ready   = lspmac_bi_init( &(lspmac_bis
[ 8]), &(md2_status.acc11c_2), 0x20, "ETEL Ready",
    "ETEL Not Ready");
etel_on      = lspmac_bi_init( &(lspmac_bis
[ 9]), &(md2_status.acc11c_2), 0x40, "ETEL On",
    "ETEL Off");
etel_init_ok = lspmac_bi_init( &(lspmac_bis
[10]), &(md2_status.acc11c_2), 0x80, "ETEL Init OK",
    "ETEL Init Not OK");
minikappa_ok = lspmac_bi_init( &(lspmac_bis
[11]), &(md2_status.acc11c_3), 0x01, "Minikappa OK",
    "Minikappa Not OK");
smart_mag_on = lspmac_bi_init( &(lspmac_bis
[12]), &(md2_status.acc11c_3), 0x04, "Smart Magnet On",
    "Smart Magnet Not On");
arm_parked   = lspmac_bi_init( &(lspmac_bis
[13]), &(md2_status.acc11c_3), 0x08, "Arm Parked",
    "Arm Not Parked");
smart_mag_err = lspmac_bi_init( &(lspmac_bis
[14]), &(md2_status.acc11c_3), 0x10, "Smart Magnet Error",
    "Smart Magnet OK");
shutter_open = lspmac_bi_init( &(lspmac_bis
[15]), &(md2_status.acc11c_3), 0x100, "Shutter Open",
    "Shutter Not Open");
smart_mag_off = lspmac_bi_init( &(lspmac_bis
[16]), &(md2_status.acc11c_5), 0x01, "Smart Magnet Off",
    "Smart Magnet Not Off");

//
// Initialize several commands that get called, perhaps, alot
//

rr_cmd.RequestType = VR_UPLOAD;
rr_cmd.Request      = VR_PMAC_READREADY;
rr_cmd.wValue       = 0;
rr_cmd.wIndex       = 0;
rr_cmd.wLength      = htons(2);
memset( rr_cmd.bData, 0, sizeof(rr_cmd.bData));

gb_cmd.RequestType = VR_UPLOAD;
gb_cmd.Request      = VR_PMAC_GETBUFFER;
gb_cmd.wValue       = 0;
gb_cmd.wIndex       = 0;

```



```

gb_cmd.wLength      = htons(1400);
memset( gb_cmd.bData, 0, sizeof(gb_cmd.bData));

cr_cmd.RequestType  = VR_UPLOAD;
cr_cmd.Request      = VR_CTRL_RESPONSE;
cr_cmd.wValue       = 0;
cr_cmd.wIndex       = 0;
cr_cmd.wLength      = htons(1400);
memset( cr_cmd.bData, 0, sizeof(cr_cmd.bData));

//
// Initialize some mutexs and conditions
//

pthread_mutex_init( &pmac_queue_mutex, &mutex_initializer);
pthread_cond_init(  &pmac_queue_cond, NULL);

lspmac_shutter_state = 0; //
    assume the shutter is now closed: not a big deal if we are wrong
pthread_mutex_init( &lspmac_shutter_mutex, &
    mutex_initializer);
pthread_cond_init(  &lspmac_shutter_cond, NULL);
pmacfd.fd = -1;

pthread_mutex_init( &lspmac_moving_mutex, &
    mutex_initializer);
pthread_cond_init(  &lspmac_moving_cond, NULL);

pthread_mutex_init( &lspmac_ascii_mutex, &mutex_initializer
    );

pthread_mutex_init( &lspmac_ascii_buffers_mutex, &
    mutex_initializer);

//
// clear the ascii communications buffers
//
{
    uint32_t cc;
    cc = 0;
    lspmac_send_command( VR_UPLOAD, VR_PMAC_SETMEM
        , 0x0e9e, 0, 4, (char *)&cc, NULL, 1, NULL);

    cc = 0x18;
    lspmac_send_command( VR_UPLOAD, VR_PMAC_SETMEM
        , 0x0e9e, 0, 4, (char *)&cc, NULL, 1, NULL);
}

lspmac_SockSendDPLine( NULL, "I5=0");
lspmac_SockSendDPLine( NULL, "ENABLE PLCC 0,2");
lspmac_SockSendDPLine( NULL, "DISABLE PLCC 1");
lspmac_SockSendDPLine( NULL, "I5=3");
}

```

7.12.4.34 int lspmac_jogabs_queue(lspmac_motor_t *, double)

Use jog to move motor to requested position.

Definition at line 2871 of file lspmac.c.

```

{

return lspmac_move_or_jog_abs_queue( mp,
    requested_position, 1);
}

```

7.12.4.35 int lspmac_move_or_jog_abs_queue(lspmac_motor_t * mp, double requested_position, int use_jo)

Move method for normal stepper and servo motor objects Returns non-zero on abort, zero if OK.

< format string for coordinate system move

< coordinate system bit

< the requested position in units of "counts"

< motor and coordinate system;

< our axis

Parameters

in	<i>mp</i>	The motor to move
in	<i>requested_position</i>	Where to move it
in	<i>use_jo</i>	1 to force jog, 0 for motion prog

Definition at line 2664 of file lspmac.c.

```

{
    char *fmt;
    int q100;
    int requested_pos_cnts;
    int coord_num, motor_num;
    char *axis;
    double u2c;
    double neutral_pos;
    double min_pos, max_pos;
    int pos_limit_hit, neg_limit_hit;
    struct timespec timeout, now;
    int err;

    pthread_mutex_lock( &(mp->mutex));

    u2c          = lsredis_getd( mp->u2c);
    motor_num    = lsredis_getl( mp->motor_num);
    coord_num    = lsredis_getl( mp->coord_num);
    axis         = lsredis_getstr( mp->axis);
    neutral_pos  = lsredis_getd( mp->neutral_pos);
    min_pos     = lsredis_getd( mp->min_pos);
    max_pos     = lsredis_getd( mp->max_pos);
    pos_limit_hit = lsredis_getd( mp->pos_limit_hit);
    neg_limit_hit = lsredis_getd( mp->neg_limit_hit);

    if( u2c == 0.0 || requested_position < min_pos || requested_position >
        max_pos) {
        //
        // Shouldn't try moving a motor that's in trouble
        //
        pthread_mutex_unlock( &(mp->mutex));
        lslogging_log_message( "lspmac_move_or_jog_abs_queue:
            %s u2c=%f requested position=%f min allowed=%f max allowed=%f", mp->name
            , u2c, requested_position, min_pos, max_pos);
        lsevents_send_event( "%s Move Aborted", mp->name);
        return 1;
    }

    if( (neg_limit_hit && (requested_position < mp->position)) || (pos_limit_hit
        && (requested_position > mp->position))) {
        pthread_mutex_unlock( &(mp->mutex));
        lslogging_log_message( "lspmac_move_or_jog_abs_queue:
            %s Moving wrong way on limit: requested position=%f current position=%f low
            limit=%d high limit=%d",
            mp->name, requested_position, mp->position
            , neg_limit_hit, pos_limit_hit);
        lsevents_send_event( "%s Move Aborted", mp->name);
        return 2;
    }

    mp->requested_position = requested_position;
    mp->not_done          = 1;
    mp->motion_seen       = 0;
    mp->command_sent      = 0;
    mp->requested_pos_cnts = u2c * (requested_position +
        neutral_pos);
    requested_pos_cnts = mp->requested_pos_cnts;

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

    pthread_mutex_unlock( &(mp->mutex));

    if( !use_jog) {

```

```

//
// Make sure the coordinate system is not moving something, wait if it is
//
pthread_mutex_lock( &lspmac_moving_mutex);

clock_gettime( CLOCK_REALTIME, &now);
//
// TODO: Have all moves estimate how long they'll take and use that here
//
timeout.tv_sec = now.tv_sec + 60.0;          // a long timeout, but
// we might really be moving something that takes this long (or longer)
timeout.tv_nsec = now.tv_nsec;

err = 0;
while( err == 0 && (lspmac_moving_flags & q100) != 0)
    err = pthread_cond_timedwait( &lspmac_moving_cond, &
        lspmac_moving_mutex, &timeout);

pthread_mutex_unlock( &lspmac_moving_mutex);

if( err == ETIMEDOUT) {
    lslogging_log_message( "
    lspmac_move_or_jog_abs_queue: Done.  lspmac_moving_flags = %0x", lspmac_moving_flags);
    lsevents_send_event( "%s Move Aborted", mp->name);
    return 1;
}

//
// Set the "we are moving this coordinate system" flag
//
lspmac_SockSendDpline( NULL, "M5075=(M5075 | %d)",
    q100);

switch( *axis) {
case 'A':
    fmt = "%d Q16=%d Q100=%d B146R";
    break;

case 'B':
    fmt = "%d Q17=%d Q100=%d B147R";
    break;

case 'C':
    fmt = "%d Q18=%d Q100=%d B148R";
    break;
case 'X':
    fmt = "%d Q10=%d Q100=%d B140R";
    break;

case 'Y':
    fmt = "%d Q11=%d Q100=%d B141R";
    break;

case 'Z':
    fmt = "%d Q12=%d Q100=%d B142R";
    break;

case 'U':
    fmt = "%d Q13=%d Q100=%d B143R";
    break;

case 'V':
    fmt = "%d Q14=%d Q100=%d B144R";
    break;

case 'W':
    fmt = "%d Q15=%d Q100=%d B145R";
    break;
}

//
// Make sure the flag has been seen
//

clock_gettime( CLOCK_REALTIME, &now);
timeout.tv_sec = now.tv_sec + 4.0;          // also a long timeout.
// This should really only take a few milliseconds on a slow day
timeout.tv_nsec = now.tv_nsec;

pthread_mutex_lock( &lspmac_moving_mutex);

err = 0;
while( err == 0 && (lspmac_moving_flags & q100) == 0)
    err = pthread_cond_timedwait( &lspmac_moving_cond, &
        lspmac_moving_mutex, &timeout);
pthread_mutex_unlock( &lspmac_moving_mutex);

```

```

    if( err == ETIMEDOUT) {
        lslogging_log_message( "
        lspmac_move_or_jog_abs_queue: Did not see flag propagate.  Move aborted.  lspmac_moving_flags = %0x",
        lspmac_moving_flags);
        lsevents_send_event( "%s Move Aborted", mp->name);
        return 1;
    }
}

pthread_mutex_lock( &(mp->mutex));
if( use_jog) {
    lspmac_SockSendDpline( mp->name, "%d j=%d",
        motor_num, requested_pos_cnts);
} else {
    lspmac_SockSendDpline( mp->name, fmt, coord_num,
        requested_pos_cnts, q100);
}
pthread_mutex_unlock( &(mp->mutex));

free( axis);

return 0;
}

```

7.12.4.36 int lspmac_move_or_jog_preset_queue (lspmac_motor_t *, char *, int)

move using a preset value returns 0 on success, non-zero on error

Definition at line 2833 of file lspmac.c.

```

{
    double pos;
    int err;
    int rtn;

    if( preset == NULL || *preset == 0) {
        lsevents_send_event( "%s Move Aborted", mp->name);
        return 0;
    }

    err = lsredis_find_preset( mp->name, preset, &pos);

    if( err != 0)
        rtn = lspmac_move_or_jog_abs_queue( mp, pos,
            use_jog);
    else {
        lsevents_send_event( "%s Move Aborted", mp->name);
        rtn = 1;
    }
    return rtn;
}

```

7.12.4.37 void lspmac_move_or_jog_queue (lspmac_motor_t *, double, int)

7.12.4.38 int lspmac_move_preset_queue (lspmac_motor_t * mp, char * preset_name)

Move a given motor to one of its preset positions.

No movement if the preset is not found.

Parameters

<i>mp</i>	lspmac motor pointer
<i>preset_name</i>	Name of the preset to use

Definition at line 2351 of file lspmac.c.

```

{
    double pos;
    int err;

    lslogging_log_message( "lspmac_move_preset_queue: Called

```

```

        with motor %s and preset named '%s'", mp->name, preset_name);

err = lsredis_find_preset( mp->name, preset_name, &pos
    );
if( err == 0)
    return 1;

err = mp->moveAbs( mp, pos);
if( !err)
    lslogging_log_message( "lspmac_move_preset_queue:
        moving %s to preset '%s' (%f)", mp->name, preset_name, pos);
//
// the abort event should have been sent in moveAbs
//
return err;
}

```

7.12.4.39 int lspmac_moveabs_queue(lspmac_motor_t *, double)

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

Definition at line 2861 of file lspmac.c.

```

{

return lspmac_move_or_jog_abs_queue( mp,
    requested_position, 0);
}

```

7.12.4.40 int lspmac_moveabs_wait(lspmac_motor_t * mp, double timeout_secs)

Wait for motor to finish moving.

Assume motion already queued, now just wait

Parameters

<i>mp</i>	The motor object to wait for
<i>timeout_secs</i>	The number of seconds to wait for. Fractional values fine.

Definition at line 2886 of file lspmac.c.

```

{

struct timespec timeout, now;
double isecs, fsecs;
int err;

//
// Copy the queue item for the most recent move request
//
clock_gettime( CLOCK_REALTIME, &now);

fsecs = modf( timeout_secs, &isecs);

timeout.tv_sec = now.tv_sec + (long)floor( isecs);
timeout.tv_nsec = now.tv_nsec + (long)floor( fsecs * 1.0e9);

timeout.tv_sec += timeout.tv_nsec / 1000000000;
timeout.tv_nsec %= 1000000000;

err = 0;
pthread_mutex_lock( &(mp->mutex));

while( err == 0 && mp->command_sent == 0)
    err = pthread_cond_timedwait( &mp->cond, &mp->mutex, &timeout);
pthread_mutex_unlock( &(mp->mutex));
if( err != 0) {
    if( err != ETIMEDOUT) {
        lslogging_log_message( "lspmac_moveabs_wait:
            unexpected error from timedwait: %d tv_sec %ld tv_nsec %ld", err, timeout.tv_sec,
            timeout.tv_nsec);
    }
    return 1;
}
}

```

```

}

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

err = 0;
pthread_mutex_lock( &(mp->mutex));
while( err == 0 && mp->motion_seen == 0)
    err = pthread_cond_timedwait( &(mp->cond), &(mp->mutex), &timeout)
    ;

if( err != 0) {
    if( err != ETIMEDOUT) {
        lslogging_log_message( "lspmac_moveabs_wait:
            unexpected error from timedwait: %d  tv_sec %ld  tv_nsec %ld", err, timeout.tv_sec,
            timeout.tv_nsec);
    }
    pthread_mutex_unlock( &(mp->mutex));
    return 1;
}

//
// wait for the motion that we know has started to finish
//
err = 0;
while( err == 0 && mp->not_done)
    err = pthread_cond_timedwait( &(mp->cond), &(mp->mutex), &timeout)
    ;

if( err != 0) {
    if( err != ETIMEDOUT) {
        lslogging_log_message( "lspmac_moveabs_wait:
            unexpected error from timedwait: %d  tv_sec %ld  tv_nsec %ld", err, timeout.tv_sec,
            timeout.tv_nsec);
    }
    pthread_mutex_unlock( &(mp->mutex));
    return 1;
}

//
// if return code was not 0 then we know we shouldn't wait for not_done flag.
// In this case the motion ended before we read the status registers
//
pthread_mutex_unlock( &(mp->mutex));
return 0;
}

```

7.12.4.41 void lspmac_run ()

Start up the lspmac thread.

Definition at line 3511 of file lspmac.c.

```

{
    char **inits;
    lspmac_motor_t *mp;
    char evts[64];
    int i;
    int active;

    pthread_create( &pmac_thread, NULL, lspmac_worker,
        NULL);

    lsevents_add_listener( "CryoSwitchChanged",
        lspmac_cryoSwitchChanged_cb);
    lsevents_add_listener( "scint In Position",
        lspmac_scint_inPosition_cb);
    lsevents_add_listener( "scintDried",
        lspmac_scint_dried_cb);
    lsevents_add_listener( "backLight 1",
        lspmac_backLight_up_cb);
    lsevents_add_listener( "backLight 0",
        lspmac_backLight_down_cb);
    lsevents_add_listener( "cam.zoom In Position",
        lspmac_light_zoom_cb);

    for( i=0; i<lspmac_nmotors; i++) {
        snprintf( evts, sizeof( evts)-1, "%s command accepted", lspmac_motors
            [i].name);
    }
}

```

```

    evts[sizeof(evts)-1] = 0;
    lsevents_add_listener( evts, lspmac_command_done_cb
    );
}

lspmac_zoom_lut_setup();
lspmac_flight_lut_setup();
lspmac_blight_lut_setup();
lspmac_fscint_lut_setup();

//
// Clear the command interfaces
//
lspmac_SockSendControlCharPrint( NULL, '\x18'
    );
{
    uint32_t cc;
    cc = 0;
    lspmac_send_command( VR_UPLOAD, VR_PMAC_SETMEM
        , 0x0e9e, 0, 4, (char *)&cc, NULL, 1, NULL);

    cc = 0x18;
    lspmac_send_command( VR_UPLOAD, VR_PMAC_SETMEM
        , 0x0e9e, 0, 4, (char *)&cc, NULL, 1, NULL);
}

//
// Initialize the MD2 pmac (ie, turn on the right plcc's etc)
//
/*
for( inits = lsredis_get_string_array(lspmac_md2_init); *inits != NULL;
    inits++) {
    lspmac_SockSendDPLine( NULL, *inits);
}
*/

//
// Initialize the pmac's support for each motor
// (ie, set the various flag for when a motor is active or not)
//
for( i=0; i<lspmac_nmotors; i++) {
    mp = &(lspmac_motors[i]);
    active = lsredis_getb( mp->active);

    // if there is a problem with "active" then don't do anything
    // On the other hand, various combinations of yes/no true/false 1/0 should
    // work
    //
    switch( active) {
    case 1:
        inits = lsredis_get_string_array( mp->active_init
        );
        break;

    case 0:
        inits = lsredis_get_string_array( mp->
        inactive_init);
        break;

    default:
        lslogging_log_message( "lspmac_run: motor %s is
        neither active nor inactive (!?)", mp->name);
        inits = NULL;
    }
    if( inits != NULL) {
        while( *inits != NULL) {
            lspmac_SockSendDPLine( NULL, *inits);
            inits++;
        }
    }
}
}
}

```

7.12.4.42 void lspmac_SockSendDPLine (char *, char * *fmt*, ...)

prepare (queue up) a line to send the dpram ascii command interface

Definition at line 1949 of file lspmac.c.

{

```

va_list arg_ptr;
uint32_t index;
char *pl;

pthread_mutex_lock( &lspmac_ascii_mutex);
index = lspmac_dpascii_on++ % LSPMAC_DPASCII_QUEUE_LENGTH
;

pl = lspmac_dpascii_queue[index].pl;

va_start( arg_ptr, fmt);
vsnprintf( pl, 159, fmt, arg_ptr);
pl[159] = 0;
va_end( arg_ptr);

lspmac_dpascii_queue[index].event = event;

pthread_mutex_unlock( &lspmac_ascii_mutex);
}

```

7.12.4.43 `lspmac_cmd_queue_t* lspmac_SockSendline (char * event, char * fmt, ...)`

Send a one line command.

Uses printf style arguments.

Parameters

in	<i>event</i>	base name for events
in	<i>fmt</i>	Printf style format string

Definition at line 1060 of file `lspmac.c`.

```

{
va_list arg_ptr;
char payload[1400];

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

lslogging_log_message( payload);

return lspmac_send_command( VR_DOWNLOAD,
    VR_PMAC_SENDLINE, 0, 0, strlen( payload), payload,
    lspmac_GetShortReplyCB, 0, event);
}

```

7.12.4.44 `void lspmac_video_rotate (double secs)`

Special motion program to collect centering video.

Definition at line 2568 of file `lspmac.c`.

```

{
double q10;           // starting position (counts)
double q11;           // delta counts
double q12;           // milliseconds to run over delta

double u2c;
double neutral_pos;

if( secs <= 0.0)
    return;

omega_zero_search = 1;

pthread_mutex_lock( &(omega->mutex));
u2c = lsredis_getd( omega->u2c);
neutral_pos = lsredis_getd( omega->neutral_pos);

q10 = neutral_pos * u2c;

```



```

q11 = 360.0 * u2c;
q12 = 1000 * secs;

omega_zero_velocity = 360.0 * u2c / secs; //
    counts/second to back calculate zero crossing time

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

```

7.12.4.45 int lsredis_cmpnstr (lsredis_obj_t * p, char * s, int n)

Definition at line 236 of file lsredis.c.

```

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

rtn = strncmp( p->value, s, n);
pthread_mutex_unlock( &p->mutex);
return rtn;
}

```

7.12.4.46 int lsredis_cmpstr (lsredis_obj_t * p, char * s)

Definition at line 225 of file lsredis.c.

```

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

rtn = strcmp( p->value, s);
pthread_mutex_unlock( &p->mutex);
return rtn;
}

```

7.12.4.47 int lsredis_find_preset (char * base, char * preset_name, double * dval)

Definition at line 756 of file lsredis.c.

```

{
char s[512];
int i;
int err;
ENTRY htab_input, *htab_output;
lsredis_obj_t *p;

i = 0;
for( i=0; i<1024; i++) {
    snprintf( s, sizeof( s)-1, "%s.%s.presets.%d.name", lsredis_head
        , base, i);
    s[sizeof(s)-1] = 0;
    htab_input.key = s;
    htab_input.data = NULL;
    err = hsearch_r( htab_input, FIND, &htab_output, &lsredis_htab)
        ;
    if( err == 0) {
        // We've run out of names to look for: done
        lslogging_log_message( "lsredis_find_preset: no
            preset for motor %s named '%s'", base, preset_name);
        *dval = 0.0;
        return 0;
    }

    // Check if we have a match
    p = htab_output->data;
}

```

```

if( lsredis_cmpstr( p, preset_name)==0) {
    // got a match, now look for the position
    snprintf( s, sizeof( s)-1, "%s.%s.preset.%d.position", lsredis_head
, base, i);
    s[sizeof(s)-1] = 0;
    htab_input.key = s;
    htab_input.data = NULL;
    err = hsearch_r( htab_input, FIND, &htab_output, &lsredis_htab
);
    if( err == 0) {
        // Name but not position? odd.
        lslogging_log_message( "lsredis_find_preset:
Error, motor %s preset '%s' has no position defined", base, preset_name);
        *dval = 0.0;
        return 0;
    }
    p = htab_output->data;
    *dval = lsredis_getd( p);
    return 1;
}
}
// How'd we get here?
// did someone really define that many presets? And then looked for one
// that's not there?
*dval = 0;
return 0;
}

```

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

Definition at line 524 of file lsredis.c.

```

{
    lsredis_obj_t *rtn;
    va_list arg_ptr;
    char k[512];
    char *kp;
    int nkp;

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

    nkp = strlen(k) + strlen( lsredis_head) + 16;           // 16
        is overkill. I know. Get over it.
    kp = calloc( nkp, sizeof( char));
    if( kp == NULL) {
        lslogging_log_message( "lsredis_get_obj: Out of memory
");
        exit( -1);
    }

    snprintf( kp, nkp-1, "%s.%s", lsredis_head, k);
    kp[nkp-1] = 0;

    pthread_mutex_lock( &lsredis_mutex);
    while( lsredis_running == 0)
        pthread_cond_wait( &lsredis_cond, &lsredis_mutex);

    rtn = _lsredis_get_obj( kp);
    pthread_mutex_unlock( &lsredis_mutex);

    free( kp);
    return rtn;
}

```

7.12.4.49 char** lsredis_get_string_array(lsredis_obj_t *p)

Definition at line 365 of file lsredis.c.

```

{
    char **rtn;

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

```

```

    rtn = p->avalue;
    pthread_mutex_unlock( &p->mutex);

    return rtn;
}

```

7.12.4.50 int lsredis_getb (lsredis_obj_t * p)

Definition at line 378 of file lsredis.c.

```

{
    int rtn;

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

    rtn = p->bvalue;
    pthread_mutex_unlock( &p->mutex);

    return rtn;
}

```

7.12.4.51 double lsredis_getd (lsredis_obj_t * p)

Definition at line 339 of file lsredis.c.

```

{
    double rtn;

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

    rtn = p->dvalue;
    pthread_mutex_unlock( &p->mutex);

    return rtn;
}

```

7.12.4.52 long int lsredis_getl (lsredis_obj_t * p)

Definition at line 352 of file lsredis.c.

```

{
    long int rtn;

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

    rtn = p->lvalue;
    pthread_mutex_unlock( &p->mutex);

    return rtn;
}

```

7.12.4.53 char* lsredis_getstr (lsredis_obj_t * p)

return a copy of the key's string value

Definition at line 263 of file lsredis.c.

```

    {
    char *rtn;

    //
    // Have to use strdup since we cannot guarantee that p->value won't be freed
    // while the caller is still using it
    //
    pthread_mutex_lock( &p->mutex);
    while( p->valid == 0)
        pthread_cond_wait( &p->cond, &p->mutex);

    rtn = strdup(p->value);
    pthread_mutex_unlock( &p->mutex);
    return rtn;
}

```

7.12.4.54 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 810 of file lsredis.c.

```

    {
    int err;
    int nerrmsg;
    char *errmsg;

    //
    // set up hash map to store redis objects
    //
    err = hcreate_r( 8192, &lsredis_hstab);
    if( err == 0) {
        lslogging_log_message( "lsredis_init: Cannot create
            hash table. Really bad things are going to happen. hcreate_r returned %d", err);
    }

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

    pthread_cond_init( &lsredis_cond, NULL);

    subac = redisAsyncConnect("127.0.0.1", 6379);
    if( subac->err) {
        lslogging_log_message( "Error: %s", subac->errstr
            );
    }

    subfd.fd = subac->c.fd;
    subfd.events = 0;
    subac->ev.data = &subfd;
    subac->ev.addRead = lsredis_addRead;
    subac->ev.delRead = lsredis_delRead;
    subac->ev.addWrite = lsredis_addWrite;
    subac->ev.delWrite = lsredis_delWrite;
    subac->ev.cleanup = lsredis_cleanup;

    roac = redisAsyncConnect("127.0.0.1", 6379);
    if( roac->err) {
        lslogging_log_message( "Error: %s", roac->errstr);
    }

    rofd.fd = roac->c.fd;
    rofd.events = 0;
    roac->ev.data = &rofd;
    roac->ev.addRead = lsredis_addRead;
    roac->ev.delRead = lsredis_delRead;
    roac->ev.addWrite = lsredis_addWrite;
    roac->ev.delWrite = lsredis_delWrite;
    roac->ev.cleanup = lsredis_cleanup;

    //wrac = redisAsyncConnect("10.1.0.3", 6379);

```

```

wrac = redisAsyncConnect("127.0.0.1", 6379);
if( wrac->err) {
    lslogging_log_message( "Error: %s", wrac->errstr);
}

wrfd.fd          = wrac->c.fd;
wrfd.events      = 0;
wrac->ev.data     = &wrfd;
wrac->ev.addRead  = lsredis_addRead;
wrac->ev.delRead  = lsredis_delRead;
wrac->ev.addWrite = lsredis_addWrite;
wrac->ev.delWrite = lsredis_delWrite;
wrac->ev.cleanup  = lsredis_cleanup;

err = regcomp( &lsredis_key_select_regex, re,
    REG_EXTENDED);
if( err != 0) {
    nerrmsg = regerror( err, &lsredis_key_select_regex,
        NULL, 0);
    if( nerrmsg > 0) {
        errmsg = calloc( nerrmsg, sizeof( char));
        nerrmsg = regerror( err, &lsredis_key_select_regex
            , errmsg, nerrmsg);
        lslogging_log_message( "lsredis_select: %s", errmsg);
        free( errmsg);
    }
}
}

```

7.12.4.55 int lsredis_regexec (const regex_t * preg, lsredis_obj_t * p, size_t nmatch, regmatch_t * pmatch, int eflags)

Definition at line 247 of file lsredis.c.

```

{
    int rtn;

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

    rtn = regexec( preg, p->value, nmatch, pmatch, eflags);

    pthread_mutex_unlock( &p->mutex);

    return rtn;
}

```

7.12.4.56 void lsredis_run ()

Definition at line 1014 of file lsredis.c.

```

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

```

7.12.4.57 void lsredis_setstr (lsredis_obj_t * p, char * fmt, ...)

Set the value and update redis.

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

redisAsyncCommandArgv used instead of redisAsyncCommand 'cause it's easier (and possible) to deal with strings that would otherwise cause hiredis to emit a bad command, like those containing spaces. < up the count of times we need to see ourselves published before we start listening to others again

< Unlock to prevent deadlock in case the service routine needs to set our value

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

Definition at line 288 of file lsredis.c.

```

{
    va_list arg_ptr;
    char v[512];
    char *argv[4];

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

    pthread_mutex_lock( &p->mutex);

    //
    // Don't send an update if a good value has not changed
    //
    if( p->valid && strcmp( v, p->value) == 0) {
        // nothing to do
        pthread_mutex_unlock( &p->mutex);
        return;
    }

    p->wait_for_me++;
    pthread_mutex_unlock( &p->mutex);

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

    pthread_mutex_lock( &lsredis_mutex);
    while( lsredis_running == 0)
        pthread_cond_wait( &lsredis_cond, &lsredis_mutex);

    redisAsyncCommand( wrac, NULL, NULL, "MULTI");
    redisAsyncCommandArgv( wrac, NULL, NULL, 4, (const char **)argv, NULL);

    redisAsyncCommand( wrac, NULL, NULL, "PUBLISH %s %s", lsredis_publisher
        , p->key);
    redisAsyncCommand( wrac, NULL, NULL, "EXEC");
    pthread_mutex_unlock( &lsredis_mutex);

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

```

7.12.4.58 void ltimer_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 ltimer.c.

```

{
    int i;
    struct timespec now;

    // Time we were called. Delay is based on call time, not queued time

```

```

//
clock_gettime( CLOCK_REALTIME, &now);

pthread_mutex_lock( &lstimer_mutex);

for( i=0; i<LSTIMER_LIST_LENGTH; i++) {
    if( lstimer_list[i].shots == 0)
        break;
}

if( i == LSTIMER_LIST_LENGTH) {
    pthread_mutex_unlock( &lstimer_mutex);

    lslogging_log_message( "lstimer_add_timer: out of
        timers for event: %s, shots: %d, secs: %u, nsecs: %u",
        event, shots, secs, nsecs);

    return;
}

strncpy( lstimer_list[i].event, event, LSEVENTS_EVENT_LENGTH
    - 1);
lstimer_list[i].event[LSEVENTS_EVENT_LENGTH
    - 1] = 0;
lstimer_list[i].shots      = shots;
lstimer_list[i].delay_secs = secs;
lstimer_list[i].delay_nsecs = nsecs;

lstimer_list[i].next_secs    = secs + now.tv_sec + (
    now.tv_nsec + nsecs) / 1000000000;
lstimer_list[i].next_nsecs   = (now.tv_nsec + nsecs
    ) % 1000000000;
lstimer_list[i].last_secs    = 0;
lstimer_list[i].last_nsecs   = 0;

lstimer_list[i].ncalls       = 0;
lstimer_list[i].init_secs    = now.tv_sec;
lstimer_list[i].init_nsecs   = now.tv_nsec;

if( shots != 0) {
    lstimer_active_timers++;
    new_timer++;
}

pthread_cond_signal( &lstimer_cond);
pthread_mutex_unlock( &lstimer_mutex);
}

```

7.12.4.59 void lstimer_init ()

Initialize the timer list and pthread stuff.

Definition at line 259 of file lstimer.c.

```

{
    int i;

    for( i=0; i<LSTIMER_LIST_LENGTH; i++) {
        lstimer_list[i].shots = 0;
    }

    pthread_mutex_init( &lstimer_mutex, NULL);
    pthread_cond_init( &lstimer_cond, NULL);
}

```

7.12.4.60 void lstimer_run ()

Start up our thread.

Definition at line 273 of file lstimer.c.

```

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

```

7.12.4.61 void lsupdate_init ()

7.12.4.62 void lsupdate_run ()

7.12.4.63 void md2cmds_init ()

Initialize the md2cmds module.

Definition at line 1356 of file md2cmds.c.

```

{
    pthread_mutexattr_t mutex_initializer;

    pthread_mutexattr_init( &mutex_initializer);
    pthread_mutexattr_settype( &mutex_initializer, PTHREAD_MUTEX_RECURSIVE);

    pthread_mutex_init( &md2cmds_mutex, &mutex_initializer);
    pthread_cond_init( &md2cmds_cond, NULL);

    pthread_mutex_init( &md2cmds_moving_mutex, &
        mutex_initializer);
    pthread_cond_init( &md2cmds_moving_cond, NULL);

    pthread_mutex_init( &md2cmds_homing_mutex, &
        mutex_initializer);
    pthread_cond_init( &md2cmds_homing_cond, NULL);

    md2cmds_md_status_code = lsredis_get_obj
        ( "md2_status_code");
    lsredis_setstr( md2cmds_md_status_code, "
        7");
}

```

7.12.4.64 void md2cmds_run ()

Start up the thread.

Definition at line 1378 of file md2cmds.c.

```

{
    pthread_create( &md2cmds_thread, NULL,
        md2cmds_worker, NULL);
    lsevents_add_listener( "omega crossed zero",
        md2cmds_rotate_cb);
    lsevents_add_listener( "omega In Position",
        md2cmds_maybe_rotate_done_cb);
    lsevents_add_listener( ".+ (Moving|In Position)",
        md2cmds_maybe_done_moving_cb);
    lsevents_add_listener( "(.+) (Homing|Homed)",
        md2cmds_maybe_done_homing_cb);
    lsevents_add_listener( "capz (Moving|In Position)",
        md2cmds_time_capz_cb);
    lsevents_add_listener( "Coordsys 1 Stopped",
        md2cmds_coordsys_1_stopped_cb);
    lsevents_add_listener( "Coordsys 2 Stopped",
        md2cmds_coordsys_2_stopped_cb);
    lsevents_add_listener( "Coordsys 3 Stopped",
        md2cmds_coordsys_3_stopped_cb);
    lsevents_add_listener( "Coordsys 4 Stopped",
        md2cmds_coordsys_4_stopped_cb);
    lsevents_add_listener( "Coordsys 5 Stopped",
        md2cmds_coordsys_5_stopped_cb);
    lsevents_add_listener( "Coordsys 7 Stopped",
        md2cmds_coordsys_7_stopped_cb);
}

```

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

Terminal output routine ala printf.

Parameters

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

Definition at line 330 of file pgpmac.c.

```

    {
        va_list arg_ptr;

        pthread_mutex_lock( &ncurses_mutex);

        va_start( arg_ptr, fmt);
        vwprintw( term_output, fmt, arg_ptr);
        va_end( arg_ptr);

        wnoutrefresh( term_output);
        wnoutrefresh( term_input);
        doupdate();

        pthread_mutex_unlock( &ncurses_mutex);
    }

```

7.12.4.66 void PmacSockSendline (char * s)

7.12.5 Variable Documentation

7.12.5.1 lspmac_motor_t* alignx

Alignment stage X.

Definition at line 90 of file lspmac.c.

7.12.5.2 lspmac_motor_t* aligny

Alignment stage Y.

Definition at line 91 of file lspmac.c.

7.12.5.3 lspmac_motor_t* alignz

Alignment stage X.

Definition at line 92 of file lspmac.c.

7.12.5.4 lspmac_motor_t* anal

Polaroid analyzer motor.

Definition at line 93 of file lspmac.c.

7.12.5.5 lspmac_motor_t* apery

Aperture Y.

Definition at line 95 of file lspmac.c.

7.12.5.6 lspmac_motor_t* aperz

Aperture Z.

Definition at line 96 of file lspmac.c.

7.12.5.7 Ispmac_bi_t* arm_parked

(whose arm? parked where?)

Definition at line 133 of file Ispmac.c.

7.12.5.8 Ispmac_motor_t* blight

Back Light DAC.

Definition at line 107 of file Ispmac.c.

7.12.5.9 Ispmac_bi_t* blight_down

Backlight is down.

Definition at line 123 of file Ispmac.c.

7.12.5.10 Ispmac_motor_t* blight_f

Back light scale factor.

Definition at line 116 of file Ispmac.c.

7.12.5.11 Ispmac_motor_t* blight_ud

Back light Up/Down actuator.

Definition at line 111 of file Ispmac.c.

7.12.5.12 Ispmac_bi_t* blight_up

Backlight is up.

Definition at line 124 of file Ispmac.c.

7.12.5.13 Ispmac_motor_t* capy

Capillary Y.

Definition at line 97 of file Ispmac.c.

7.12.5.14 Ispmac_motor_t* capz

Capillary Z.

Definition at line 98 of file Ispmac.c.

7.12.5.15 Ispmac_motor_t* cenx

Centering Table X.

Definition at line 100 of file Ispmac.c.

7.12.5.16 Ispmac_motor_t* ceny

Centering Table Y.

Definition at line 101 of file Ispmac.c.

7.12.5.17 Ispmac_motor_t* cryo

Move the cryostream towards or away from the crystal.

Definition at line 112 of file Ispmac.c.

7.12.5.18 Ispmac_bi_t* cryo_back

cryo is in the back position

Definition at line 125 of file Ispmac.c.

7.12.5.19 Ispmac_bi_t* cryo_switch

that little toggle switch for the cryo

Definition at line 122 of file Ispmac.c.

7.12.5.20 Ispmac_motor_t* dryer

blow air on the scintillator to dry it off

Definition at line 113 of file Ispmac.c.

7.12.5.21 Ispmac_bi_t* etel_init_ok

ETEL initialized OK.

Definition at line 130 of file Ispmac.c.

7.12.5.22 Ispmac_bi_t* etel_on

ETEL is on.

Definition at line 129 of file Ispmac.c.

7.12.5.23 Ispmac_bi_t* etel_ready

ETEL is ready.

Definition at line 128 of file Ispmac.c.

7.12.5.24 Ispmac_motor_t* flight

Front Light DAC.

Definition at line 106 of file Ispmac.c.

7.12.5.25 Ispmac_motor_t* flight_f

Front light scale factor.

Definition at line 117 of file Ispmac.c.

7.12.5.26 Ispmac_motor_t* flight_oo

Turn front light on/off.

Definition at line 115 of file Ispmac.c.

7.12.5.27 Ispmac_motor_t* fluo

Move the fluorescence detector in/out.

Definition at line 114 of file Ispmac.c.

7.12.5.28 Ispmac_bi_t* fluor_back

fluor is in the back position

Definition at line 126 of file Ispmac.c.

7.12.5.29 Ispmac_motor_t* fscint

Scintillator Piezo DAC.

Definition at line 108 of file Ispmac.c.

7.12.5.30 Ispmac_motor_t* fshut

Fast shutter.

Definition at line 105 of file Ispmac.c.

7.12.5.31 Ispmac_bi_t* hp_air

High pressure air OK.

Definition at line 121 of file Ispmac.c.

7.12.5.32 Ispmac_motor_t* kappa

Kappa.

Definition at line 102 of file Ispmac.c.

7.12.5.33 Ispmac_bi_t* lp_air

Low pressure air OK.

Definition at line 120 of file Ispmac.c.

7.12.5.34 `lspg_demandairrights_t` `lspg_demandairrights`

our demandairrights object

Definition at line 66 of file `lspg.c`.

7.12.5.35 `lspg_getcenter_t` `lspg_getcenter`

the getcenter object

Definition at line 65 of file `lspg.c`.

7.12.5.36 `lspg_getcurrentsampleid_t` `lspg_getcurrentsampleid`

our currentsample id

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

7.12.5.37 `lspg_nextsample_t` `lspg_nextsample`

the very next sample

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

7.12.5.38 `lspg_nextshot_t` `lspg_nextshot`

the nextshot object

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

7.12.5.39 `lspg_starttransfer_t` `lspg_starttransfer`

start a sample transfer

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

7.12.5.40 `lspg_waitcryo_t` `lspg_waitcryo`

signal the robot

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

7.12.5.41 `lspmac_motor_t` `lspmac_motors[]`

All our motors.

Definition at line 87 of file `lspmac.c`.

7.12.5.42 `pthread_cond_t` `lspmac_moving_cond`

Wait for motor(s) to finish moving condition.

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

7.12.5.43 int lspmac_moving_flags

Flag used to implement motor moving condition.

Definition at line 65 of file lspmac.c.

7.12.5.44 pthread_mutex_t lspmac_moving_mutex

Coordinate moving motors between threads.

Definition at line 63 of file lspmac.c.

7.12.5.45 int lspmac_nmotors

The number of motors we manage.

Definition at line 88 of file lspmac.c.

7.12.5.46 pthread_cond_t lspmac_shutter_cond

Allows waiting for the shutter status to change.

Definition at line 62 of file lspmac.c.

7.12.5.47 int lspmac_shutter_has_opened

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

Definition at line 60 of file lspmac.c.

7.12.5.48 pthread_mutex_t lspmac_shutter_mutex

Coordinates threads reading shutter status.

Definition at line 61 of file lspmac.c.

7.12.5.49 int lspmac_shutter_state

State of the shutter, used to detect changes.

Definition at line 59 of file lspmac.c.

7.12.5.50 pthread_mutex_t md2_status_mutex

Synchronize reading/writing status buffer.

Definition at line 341 of file lspmac.c.

7.12.5.51 char md2cmds_cmd[]

our command;

Definition at line 24 of file md2cmds.c.

7.12.5.52 pthread_cond_t md2cmds_cond

condition to signal when it's time to run an md2 command

Definition at line 10 of file md2cmds.c.

7.12.5.53 Isredis_obj_t* md2cmds_md_status_code

Definition at line 26 of file md2cmds.c.

7.12.5.54 pthread_mutex_t md2cmds_mutex

mutex for the condition

Definition at line 11 of file md2cmds.c.

7.12.5.55 pthread_cond_t md2cmds_pg_cond**7.12.5.56 pthread_mutex_t md2cmds_pg_mutex****7.12.5.57 lspmac_bi_t* minikappa_ok**

Minikappa is OK (whatever that means)

Definition at line 131 of file lspmac.c.

7.12.5.58 pthread_mutex_t ncurses_mutex

allow more than one thread access to the screen

Definition at line 242 of file pgpmac.c.

7.12.5.59 lspmac_motor_t* omega

MD2 omega axis (the air bearing)

Definition at line 89 of file lspmac.c.

7.12.5.60 struct timespec omega_zero_time

Time we believe that omega crossed zero.

Definition at line 72 of file lspmac.c.

7.12.5.61 lspmac_motor_t* phi

Phi (not data collection axis)

Definition at line 103 of file lspmac.c.

7.12.5.62 pthread_cond_t pmac_queue_cond

wait for a command to be sent to PMAC before continuing

Definition at line 78 of file lspmac.c.

7.12.5.63 pthread_mutex_t pmac.queue_mutex

manage access to the pmac command queue

Definition at line 77 of file lspmac.c.

7.12.5.64 lspmac_bi_t* sample_detected

smart magnet detected sample

Definition at line 127 of file lspmac.c.

7.12.5.65 lspmac_motor_t* scint

Scintillator Z.

Definition at line 99 of file lspmac.c.

7.12.5.66 lspmac_bi_t* shutter_open

shutter is open (note in pmc says this is a slow input)

Definition at line 134 of file lspmac.c.

7.12.5.67 lspmac_bi_t* smart_mag_err

smart magnet error (coil broken perhaps)

Definition at line 135 of file lspmac.c.

7.12.5.68 lspmac_bi_t* smart_mag_off

smart magnet is off

Definition at line 136 of file lspmac.c.

7.12.5.69 lspmac_bi_t* smart_mag_on

smart magnet is on

Definition at line 132 of file lspmac.c.

7.12.5.70 lspmac_motor_t* smart_mag_oo

Smart Magnet on/off.

Definition at line 110 of file lspmac.c.

7.12.5.71 WINDOW* term.input

place to put the cursor

Definition at line 238 of file pgpmac.c.

7.12.5.72 WINDOW* term_output

place to print stuff out

Definition at line 237 of file pgpmac.c.

7.12.5.73 WINDOW* term_status

shutter, lamp, air, etc status

Definition at line 239 of file pgpmac.c.

7.12.5.74 WINDOW* term_status2

shutter, lamp, air, etc status

Definition at line 240 of file pgpmac.c.

7.12.5.75 lspmac_motor_t* zoom

Optical zoom.

Definition at line 94 of file lspmac.c.

Index

- [_GNU_SOURCE](#)
 - [pgpmac.h, 273](#)
- [__init__](#)
 - [iniParser::iniParser, 16](#)
- [_lspmac_motor_init](#)
 - [lspmac.c, 147](#)
- [_lsredis_get_obj](#)
 - [lsredis.c, 213](#)
- [_lsredis_set_value](#)
 - [lsredis.c, 214](#)
- [acc11c_1](#)
 - [md2StatusStruct, 60](#)
- [acc11c_2](#)
 - [md2StatusStruct, 60](#)
- [acc11c_3](#)
 - [md2StatusStruct, 60](#)
- [acc11c_5](#)
 - [md2StatusStruct, 60](#)
- [acc11c_6](#)
 - [md2StatusStruct, 60](#)
- [active](#)
 - [lspg_nextshot_struct, 29](#)
 - [lspmac_motor_struct, 48](#)
- [active2](#)
 - [lspg_nextshot_struct, 29](#)
- [active2_isnull](#)
 - [lspg_nextshot_struct, 30](#)
- [active_init](#)
 - [lspmac_motor_struct, 48](#)
- [active_isnull](#)
 - [lspg_nextshot_struct, 30](#)
- [actual_pos_cnts](#)
 - [lspmac_motor_struct, 48](#)
- [actual_pos_cnts_p](#)
 - [lspmac_motor_struct, 48](#)
- [addRead](#)
 - [kvredis.c, 72](#)
- [addWrite](#)
 - [kvredis.c, 72](#)
- [alignx](#)
 - [lspmac.c, 201](#)
 - [pgpmac.h, 305](#)
- [alignx_act_pos](#)
 - [md2StatusStruct, 60](#)
- [alignx_status_1](#)
 - [md2StatusStruct, 60](#)
- [alignx_status_2](#)
 - [md2StatusStruct, 60](#)
- [aligny](#)
 - [lspmac.c, 201](#)
 - [pgpmac.h, 305](#)
- [aligny_act_pos](#)
 - [md2StatusStruct, 60](#)
- [aligny_status_1](#)
 - [md2StatusStruct, 60](#)
- [aligny_status_2](#)
 - [md2StatusStruct, 60](#)
- [alignz](#)
 - [lspmac.c, 201](#)
 - [pgpmac.h, 305](#)
- [alignz_act_pos](#)
 - [md2StatusStruct, 60](#)
- [alignz_status_1](#)
 - [md2StatusStruct, 60](#)
- [alignz_status_2](#)
 - [md2StatusStruct, 61](#)
- [anal](#)
 - [lspmac.c, 201](#)
 - [pgpmac.h, 305](#)
- [analyzer_act_pos](#)
 - [md2StatusStruct, 61](#)
- [analyzer_status_1](#)
 - [md2StatusStruct, 61](#)
- [analyzer_status_2](#)
 - [md2StatusStruct, 61](#)
- [aperturey_act_pos](#)
 - [md2StatusStruct, 61](#)
- [aperturey_status_1](#)
 - [md2StatusStruct, 61](#)
- [aperturey_status_2](#)
 - [md2StatusStruct, 61](#)
- [aperturez_act_pos](#)
 - [md2StatusStruct, 61](#)
- [aperturez_status_1](#)
 - [md2StatusStruct, 61](#)
- [aperturez_status_2](#)
 - [md2StatusStruct, 61](#)
- [apery](#)
 - [lspmac.c, 201](#)
 - [pgpmac.h, 305](#)
- [aperz](#)
 - [lspmac.c, 201](#)
 - [pgpmac.h, 305](#)
- [arm_parked](#)
 - [lspmac.c, 201](#)
 - [pgpmac.h, 305](#)
- [avalue](#)
 - [lsredis_obj_struct, 54](#)

- ax
 - lspg_nextshot_struct, 30
- ax2
 - lspg_nextshot_struct, 30
- ax2_isnull
 - lspg_nextshot_struct, 30
- ax_isnull
 - lspg_nextshot_struct, 30
- axis
 - lspmac_motor_struct, 48
- ay
 - lspg_nextshot_struct, 30
- ay2
 - lspg_nextshot_struct, 30
- ay2_isnull
 - lspg_nextshot_struct, 30
- ay_isnull
 - lspg_nextshot_struct, 30
- az
 - lspg_nextshot_struct, 30
- az2
 - lspg_nextshot_struct, 31
- az2_isnull
 - lspg_nextshot_struct, 31
- az_isnull
 - lspg_nextshot_struct, 31
- b
 - mk_pgpmac_redis, 12
- bData
 - tagEthernetCmd, 66
- back_dac
 - md2StatusStruct, 61
- bi_list
 - mk_pgpmac_redis, 12
- blight
 - lspmac.c, 201
 - pgpmac.h, 306
- blight_down
 - lspmac.c, 201
 - pgpmac.h, 306
- blight_f
 - lspmac.c, 201
 - pgpmac.h, 306
- blight_ud
 - lspmac.c, 202
 - pgpmac.h, 306
- blight_up
 - lspmac.c, 202
 - pgpmac.h, 306
- bvalue
 - lsredis_obj_struct, 54
- capy
 - lspmac.c, 202
 - pgpmac.h, 306
- capy_act_pos
 - md2StatusStruct, 61
- capy_status_1
 - md2StatusStruct, 62
- capy_status_2
 - md2StatusStruct, 62
- capz
 - lspmac.c, 202
 - pgpmac.h, 306
- capz_act_pos
 - md2StatusStruct, 62
- capz_status_1
 - md2StatusStruct, 62
- capz_status_2
 - md2StatusStruct, 62
- cb
 - lsevents_listener_struct, 18
- centerx_act_pos
 - md2StatusStruct, 62
- centerx_status_1
 - md2StatusStruct, 62
- centerx_status_2
 - md2StatusStruct, 62
- centery_act_pos
 - md2StatusStruct, 62
- centery_status_1
 - md2StatusStruct, 62
- centery_status_2
 - md2StatusStruct, 62
- cenx
 - lspmac.c, 202
 - pgpmac.h, 306
- ceny
 - lspmac.c, 202
 - pgpmac.h, 306
- changeEventOff
 - lspmac_bi_struct, 43
- changeEventOn
 - lspmac_bi_struct, 43
- cleanstr
 - lspmac.c, 148
- cleanup
 - kvredis.c, 72
- cmdac
 - kvredis.c, 83
- cmdfd
 - kvredis.c, 83
- command_buf
 - lspmac_ascii_buffers_struct, 42
- command_buf_cc
 - lspmac_ascii_buffers_struct, 42
- command_sent
 - lspmac_motor_struct, 48
- command_str
 - lspmac_ascii_buffers_struct, 42
- cond
 - lspg_demandairrights_struct, 20
 - lspg_getcenter_struct, 21
 - lspg_getcurrentsampleid_struct, 23
 - lspg_lock_detector_struct, 24
 - lspg_lock_diffractionmeter_struct, 25

- lspg_nextsample_struct, 26
- lspg_nextshot_struct, 31
- lspg_seq_run_prep_struct, 38
- lspg_starttransfer_struct, 39
- lspg_wait_for_detector_struct, 40
- lspg_waitcryo_struct, 40
- lspmac_motor_struct, 49
- lsredis_obj_struct, 54
- coord_num
 - lspmac_motor_struct, 49
- cr_cmd
 - lspmac.c, 202
- cryo
 - lspmac.c, 202
 - pgpmac.h, 307
- cryo_back
 - lspmac.c, 202
 - pgpmac.h, 307
- cryo_switch
 - lspmac.c, 203
 - pgpmac.h, 307
- cvalue
 - lsredis_obj_struct, 55
- cx
 - lspg_nextshot_struct, 31
- cx2
 - lspg_nextshot_struct, 31
- cx2_isnull
 - lspg_nextshot_struct, 31
- cx_isnull
 - lspg_nextshot_struct, 31
- cy
 - lspg_nextshot_struct, 31
- cy2
 - lspg_nextshot_struct, 31
- cy2_isnull
 - lspg_nextshot_struct, 32
- cy_isnull
 - lspg_nextshot_struct, 32
- dac_mvar
 - lspmac_motor_struct, 49
- dax
 - lspg_getcenter_struct, 21
- dax_isnull
 - lspg_getcenter_struct, 21
- day
 - lspg_getcenter_struct, 21
- day_isnull
 - lspg_getcenter_struct, 22
- daz
 - lspg_getcenter_struct, 22
- daz_isnull
 - lspg_getcenter_struct, 22
- dbmem
 - lspmac.c, 203
- dbmemIn
 - lspmac.c, 203
- dcx
 - lspg_getcenter_struct, 22
- dcx_isnull
 - lspg_getcenter_struct, 22
- dcy
 - lspg_getcenter_struct, 22
- dcy_isnull
 - lspg_getcenter_struct, 22
- debugCB
 - kvredis.c, 73
- delRead
 - kvredis.c, 73
- delWrite
 - kvredis.c, 73
- delay_nsecs
 - lstimer_list_struct, 57
- delay_secs
 - lstimer_list_struct, 57
- dryer
 - lspmac.c, 203
 - pgpmac.h, 307
- dsdir
 - lspg_nextshot_struct, 32
- dsdir_isnull
 - lspg_nextshot_struct, 32
- dsdist
 - lspg_nextshot_struct, 32
- dsdist2
 - lspg_nextshot_struct, 32
- dsdist2_isnull
 - lspg_nextshot_struct, 32
- dsdist_isnull
 - lspg_nextshot_struct, 32
- dsexp
 - lspg_nextshot_struct, 32
- dsexp2
 - lspg_nextshot_struct, 32
- dsexp2_isnull
 - lspg_nextshot_struct, 33
- dsexp_isnull
 - lspg_nextshot_struct, 33
- dshpid
 - lspg_nextshot_struct, 33
- dshpid_isnull
 - lspg_nextshot_struct, 33
- dskappa
 - lspg_nextshot_struct, 33
- dskappa2
 - lspg_nextshot_struct, 33
- dskappa2_isnull
 - lspg_nextshot_struct, 33
- dskappa_isnull
 - lspg_nextshot_struct, 33
- dsnrg
 - lspg_nextshot_struct, 33
- dsnrg2
 - lspg_nextshot_struct, 33
- dsnrg2_isnull
 - lspg_nextshot_struct, 34

- dsnrg_isnull
 - lspg_nextshot_struct, [34](#)
- dsomega
 - lspg_nextshot_struct, [34](#)
- dsomega2
 - lspg_nextshot_struct, [34](#)
- dsomega2_isnull
 - lspg_nextshot_struct, [34](#)
- dsomega_isnull
 - lspg_nextshot_struct, [34](#)
- dsoscaxis
 - lspg_nextshot_struct, [34](#)
- dsoscaxis2
 - lspg_nextshot_struct, [34](#)
- dsoscaxis2_isnull
 - lspg_nextshot_struct, [34](#)
- dsoscaxis_isnull
 - lspg_nextshot_struct, [34](#)
- dsowidth
 - lspg_nextshot_struct, [34](#)
- dsowidth2
 - lspg_nextshot_struct, [35](#)
- dsowidth2_isnull
 - lspg_nextshot_struct, [35](#)
- dsowidth_isnull
 - lspg_nextshot_struct, [35](#)
- dsphi
 - lspg_nextshot_struct, [35](#)
- dsphi2
 - lspg_nextshot_struct, [35](#)
- dsphi2_isnull
 - lspg_nextshot_struct, [35](#)
- dsphi_isnull
 - lspg_nextshot_struct, [35](#)
- dspid
 - lspg_nextshot_struct, [35](#)
- dspid_isnull
 - lspg_nextshot_struct, [35](#)
- dummy1
 - md2StatusStruct, [62](#)
- dummy2
 - md2StatusStruct, [63](#)
- dummy3
 - md2StatusStruct, [63](#)
- dummy4
 - md2StatusStruct, [63](#)
- dummy5
 - md2StatusStruct, [63](#)
- dummy6
 - md2StatusStruct, [63](#)
- dummy7
 - md2StatusStruct, [63](#)
- dummy8
 - md2StatusStruct, [63](#)
- dummy9
 - md2StatusStruct, [63](#)
- dummyA
 - md2StatusStruct, [63](#)
- dummyB
 - md2StatusStruct, [63](#)
- dvalue
 - lsredis_obj_struct, [55](#)
- etel_init_ok
 - lspmac.c, [203](#)
 - pgpmac.h, [307](#)
- etel_on
 - lspmac.c, [203](#)
 - pgpmac.h, [307](#)
- etel_ready
 - lspmac.c, [203](#)
 - pgpmac.h, [307](#)
- ethCmdOff
 - lspmac.c, [203](#)
- ethCmdOn
 - lspmac.c, [203](#)
- ethCmdQueue
 - lspmac.c, [204](#)
- ethCmdReply
 - lspmac.c, [204](#)
- event
 - lspmac_cmd_queue_struct, [45](#)
 - lspmac_dpascii_queue_struct, [45](#)
 - lstimer_list_struct, [57](#)
- events_name
 - lsredis_obj_struct, [55](#)
- evp
 - lsevents_queue_struct, [19](#)
- f
 - iniParser::iniParser, [17](#)
 - mk_pgpmac_redis, [12](#)
- fd_service
 - kvredis.c, [74](#)
- first_time
 - lspmac_bi_struct, [43](#)
- flight
 - lspmac.c, [204](#)
 - pgpmac.h, [307](#)
- flight_f
 - lspmac.c, [204](#)
 - pgpmac.h, [307](#)
- flight_oo
 - lspmac.c, [204](#)
 - pgpmac.h, [308](#)
- fluo
 - lspmac.c, [204](#)
 - pgpmac.h, [308](#)
- fluor_back
 - lspmac.c, [204](#)
 - pgpmac.h, [308](#)
- fnc
 - mk_pgpmac_redis, [12](#)
- front_dac
 - md2StatusStruct, [63](#)
- fs_has_opened
 - md2StatusStruct, [63](#)

- fs_has_opened_globally
 - md2StatusStruct, 64
- fs_is_open
 - md2StatusStruct, 64
- fscint
 - lspmac.c, 204
 - pgpmac.h, 308
- fshut
 - lspmac.c, 204
 - pgpmac.h, 308
- gb_cmd
 - lspmac.c, 205
- get
 - iniParser::iniParser, 16
- getcurrentsampleid
 - lspg_getcurrentsampleid_struct, 23
- getcurrentsampleid_isnull
 - lspg_getcurrentsampleid_struct, 24
- getivars
 - lspmac.c, 205
- getmvars
 - lspmac.c, 205
- handler
 - lstimer.c, 230
- hard_ini
 - mk_pgpmac_redis, 12
- hard_ini_fields
 - mk_pgpmac_redis, 12
- has_option
 - iniParser::iniParser, 16
- has_section
 - iniParser::iniParser, 16
- head
 - mk_pgpmac_redis, 12
- hex_dump
 - lspmac.c, 148
- hi
 - mk_pgpmac_redis, 12
- hits
 - lsredis_obj_struct, 55
- home
 - lspmac_motor_struct, 49
- homing
 - lspmac_motor_struct, 49
- hp_air
 - lspmac.c, 205
 - pgpmac.h, 308
- i
 - mk_pgpmac_redis, 12
- inactive_init
 - lspmac_motor_struct, 49
- iniParser, 11
 - ip, 11
- iniParser.iniParser, 15
- iniParser.py, 69
- iniParser::iniParser
 - __init__, 16
 - f, 17
 - get, 16
 - has_option, 16
 - has_section, 16
 - options, 16
 - read, 16
 - sd, 17
 - sections, 17
- init_nsecs
 - lstimer_list_struct, 57
- init_secs
 - lstimer_list_struct, 57
- ip
 - iniParser, 11
- kappa
 - lspmac.c, 205
 - pgpmac.h, 308
- kappa_act_pos
 - md2StatusStruct, 64
- kappa_status_1
 - md2StatusStruct, 64
- kappa_status_2
 - md2StatusStruct, 64
- key
 - lsredis_obj_struct, 55
- kvredis.c, 69
 - addRead, 72
 - addWrite, 72
 - cleanup, 72
 - cmdac, 83
 - cmdfd, 83
 - debugCB, 73
 - delRead, 73
 - delWrite, 73
 - fd_service, 74
 - kvseq, 83
 - LS_PG_STATE_IDLE, 71
 - LS_PG_STATE_INIT, 71
 - LS_PG_STATE_RECV, 72
 - LS_PG_STATE_RESET, 72
 - LS_PG_STATE_SEND, 72
 - ls_pg_state, 83
 - lspg_allkvs_cb, 74
 - lspg_connectPoll_response, 84
 - lspg_flush, 75
 - lspg_next_state, 75
 - lspg_notice_processor, 76
 - lspg_pg_connect, 76
 - lspg_pg_service, 77
 - lspg_query_next, 78
 - lspg_query_push, 79
 - lspg_query_queue, 84
 - lspg_query_queue_off, 84
 - lspg_query_queue_on, 84
 - lspg_query_queue_reply, 84
 - lspg_query_queue_t, 72
 - lspg_query_reply_next, 79

- lspg_query_reply_peek, 80
- lspg_receive, 80
- lspg_resetPoll_response, 84
- lspg_send_next_query, 81
- lspgfd, 84
- main, 81
- now, 84
- q, 84
- redisDisconnectCB, 83
- subac, 85
- subfd, 85
- kvseq
 - kvredis.c, 83
- LS_PG_STATE_IDLE
 - kvredis.c, 71
 - lspg.c, 100
- LS_PG_STATE_INIT
 - kvredis.c, 71
 - lspg.c, 101
- LS_PG_STATE_RECV
 - kvredis.c, 72
 - lspg.c, 101
- LS_PG_STATE_RESET
 - kvredis.c, 72
 - lspg.c, 101
- LS_PG_STATE_SEND
 - kvredis.c, 72
 - lspg.c, 101
- LS_PMAC_STATE_CR
 - lspmac.c, 143
- LS_PMAC_STATE_GB
 - lspmac.c, 144
- LS_PMAC_STATE_GMR
 - lspmac.c, 144
- LS_PMAC_STATE_IDLE
 - lspmac.c, 144
- LS_PMAC_STATE_RESET
 - lspmac.c, 144
- LS_PMAC_STATE_RR
 - lspmac.c, 144
- LS_PMAC_STATE_SC
 - lspmac.c, 144
- LS_PMAC_STATE_WACK
 - lspmac.c, 144
- LS_PMAC_STATE_WCR
 - lspmac.c, 144
- LS_PMAC_STATE_WGB
 - lspmac.c, 144
- LSLOGGING_FILE_NAME
 - lslogging.c, 92
- LSPMAC_PRESET_REGEX
 - lspmac.c, 145
- LSTIMER_LIST_LENGTH
 - lstimer.c, 230
- last_nsecs
 - lstimer_list_struct, 57
- last_secs
 - lstimer_list_struct, 57
- lmsg
 - lslogging_queue_struct, 19
- lp_air
 - lspmac.c, 205
 - pgpmac.h, 308
- ls_pg_state
 - kvredis.c, 83
 - lspg.c, 132
- ls_pmac_state
 - lspmac.c, 205
- lsConnect
 - lspmac.c, 149
- lsevents.c, 85
 - lsevents_add_listener, 87
 - lsevents_init, 87
 - lsevents_listener_mutex, 90
 - lsevents_listener_t, 87
 - lsevents_listeners_p, 90
 - lsevents_queue, 90
 - lsevents_queue_cond, 90
 - lsevents_queue_mutex, 90
 - lsevents_queue_off, 90
 - lsevents_queue_on, 91
 - lsevents_queue_t, 87
 - lsevents_remove_listener, 88
 - lsevents_run, 88
 - lsevents_send_event, 88
 - lsevents_thread, 91
 - lsevents_worker, 89
- lsevents_add_listener
 - lsevents.c, 87
 - pgpmac.h, 275
- lsevents_init
 - lsevents.c, 87
 - pgpmac.h, 275
- lsevents_listener_mutex
 - lsevents.c, 90
- lsevents_listener_struct, 17
 - cb, 18
 - next, 18
 - raw_regexp, 18
 - re, 18
- lsevents_listener_t
 - lsevents.c, 87
- lsevents_listeners_p
 - lsevents.c, 90
- lsevents_queue
 - lsevents.c, 90
- lsevents_queue_cond
 - lsevents.c, 90
- lsevents_queue_mutex
 - lsevents.c, 90
- lsevents_queue_off
 - lsevents.c, 90
- lsevents_queue_on
 - lsevents.c, 91
- lsevents_queue_struct, 18
 - evp, 19

- lsevents_queue_t
 - lsevents.c, [87](#)
- lsevents_remove_listener
 - lsevents.c, [88](#)
 - pgpmac.h, [276](#)
- lsevents_run
 - lsevents.c, [88](#)
 - pgpmac.h, [276](#)
- lsevents_send_event
 - lsevents.c, [88](#)
 - pgpmac.h, [276](#)
- lsevents_thread
 - lsevents.c, [91](#)
- lsevents_worker
 - lsevents.c, [89](#)
- lslogging.c, [91](#)
 - LSLOGGING_FILE_NAME, [92](#)
 - lslogging_cond, [94](#)
 - lslogging_file, [94](#)
 - lslogging_init, [93](#)
 - lslogging_log_message, [93](#)
 - lslogging_mutex, [94](#)
 - lslogging_off, [95](#)
 - lslogging_on, [95](#)
 - lslogging_queue, [95](#)
 - lslogging_queue_t, [93](#)
 - lslogging_run, [94](#)
 - lslogging_thread, [95](#)
 - lslogging_worker, [94](#)
- lslogging_cond
 - lslogging.c, [94](#)
- lslogging_file
 - lslogging.c, [94](#)
- lslogging_init
 - lslogging.c, [93](#)
 - pgpmac.h, [277](#)
- lslogging_log_message
 - lslogging.c, [93](#)
 - pgpmac.h, [277](#)
- lslogging_mutex
 - lslogging.c, [94](#)
- lslogging_off
 - lslogging.c, [95](#)
- lslogging_on
 - lslogging.c, [95](#)
- lslogging_queue
 - lslogging.c, [95](#)
- lslogging_queue_struct, [19](#)
 - lmsg, [19](#)
 - ltime, [19](#)
- lslogging_queue_t
 - lslogging.c, [93](#)
- lslogging_run
 - lslogging.c, [94](#)
 - pgpmac.h, [278](#)
- lslogging_thread
 - lslogging.c, [95](#)
- lslogging_worker
 - lslogging.c, [94](#)
- lspg.c, [95](#)
 - LS_PG_STATE_IDLE, [100](#)
 - LS_PG_STATE_INIT, [101](#)
 - LS_PG_STATE_RECV, [101](#)
 - LS_PG_STATE_RESET, [101](#)
 - LS_PG_STATE_SEND, [101](#)
 - ls_pg_state, [132](#)
 - lspg_array2ptrs, [102](#)
 - lspg_cmd_cb, [103](#)
 - lspg_connectPoll_response, [132](#)
 - lspg_demandairrights, [132](#)
 - lspg_demandairrights_all, [103](#)
 - lspg_demandairrights_call, [104](#)
 - lspg_demandairrights_cb, [104](#)
 - lspg_demandairrights_init, [104](#)
 - lspg_demandairrights_wait, [104](#)
 - lspg_flush, [104](#)
 - lspg_getcenter, [132](#)
 - lspg_getcenter_all, [105](#)
 - lspg_getcenter_call, [105](#)
 - lspg_getcenter_cb, [105](#)
 - lspg_getcenter_done, [106](#)
 - lspg_getcenter_init, [106](#)
 - lspg_getcenter_wait, [107](#)
 - lspg_getcurrentsampleid, [132](#)
 - lspg_getcurrentsampleid_call, [107](#)
 - lspg_getcurrentsampleid_cb, [107](#)
 - lspg_getcurrentsampleid_init, [107](#)
 - lspg_getcurrentsampleid_read, [108](#)
 - lspg_getcurrentsampleid_wait_for_id, [108](#)
 - lspg_init, [108](#)
 - lspg_lock_detector, [132](#)
 - lspg_lock_detector_all, [109](#)
 - lspg_lock_detector_call, [109](#)
 - lspg_lock_detector_cb, [109](#)
 - lspg_lock_detector_done, [109](#)
 - lspg_lock_detector_init, [109](#)
 - lspg_lock_detector_t, [101](#)
 - lspg_lock_detector_wait, [110](#)
 - lspg_lock_diffractionmeter, [133](#)
 - lspg_lock_diffractionmeter_all, [110](#)
 - lspg_lock_diffractionmeter_call, [110](#)
 - lspg_lock_diffractionmeter_cb, [110](#)
 - lspg_lock_diffractionmeter_done, [111](#)
 - lspg_lock_diffractionmeter_init, [111](#)
 - lspg_lock_diffractionmeter_t, [101](#)
 - lspg_lock_diffractionmeter_wait, [111](#)
 - lspg_next_state, [111](#)
 - lspg_nextaction_cb, [112](#)
 - lspg_nextsample, [133](#)
 - lspg_nextsample_all, [113](#)
 - lspg_nextsample_call, [113](#)
 - lspg_nextsample_cb, [113](#)
 - lspg_nextsample_done, [114](#)
 - lspg_nextsample_init, [114](#)
 - lspg_nextsample_wait, [114](#)
 - lspg_nextshot, [133](#)

- lspg_nextshot_call, 114
- lspg_nextshot_cb, 115
- lspg_nextshot_done, 118
- lspg_nextshot_init, 119
- lspg_nextshot_wait, 119
- lspg_notice_processor, 119
- lspg_pg_connect, 119
- lspg_pg_service, 120
- lspg_query_next, 122
- lspg_query_push, 122
- lspg_query_queue, 133
- lspg_query_queue_off, 133
- lspg_query_queue_on, 133
- lspg_query_queue_reply, 133
- lspg_query_reply_next, 123
- lspg_query_reply_peek, 123
- lspg_queue_cond, 133
- lspg_queue_mutex, 133
- lspg_receive, 123
- lspg_resetPoll_response, 134
- lspg_run, 124
- lspg_send_next_query, 124
- lspg_seq_run_prep, 134
- lspg_seq_run_prep_all, 125
- lspg_seq_run_prep_call, 125
- lspg_seq_run_prep_cb, 126
- lspg_seq_run_prep_done, 126
- lspg_seq_run_prep_init, 126
- lspg_seq_run_prep_t, 101
- lspg_seq_run_prep_wait, 127
- lspg_sig_service, 127
- lspg_starttransfer, 134
- lspg_starttransfer_all, 127
- lspg_starttransfer_call, 128
- lspg_starttransfer_cb, 128
- lspg_starttransfer_done, 128
- lspg_starttransfer_init, 128
- lspg_starttransfer_wait, 128
- lspg_thread, 134
- lspg_wait_for_detector, 134
- lspg_wait_for_detector_all, 129
- lspg_wait_for_detector_call, 129
- lspg_wait_for_detector_cb, 129
- lspg_wait_for_detector_done, 129
- lspg_wait_for_detector_init, 130
- lspg_wait_for_detector_t, 101
- lspg_wait_for_detector_wait, 130
- lspg_waitcryo, 134
- lspg_waitcryo_all, 130
- lspg_waitcryo_cb, 130
- lspg_waitcryo_init, 130
- lspg_worker, 131
- lspgfd, 134
- lspmac_sample_detector_cb, 132
- now, 134
- q, 134
- lspg_allkvs_cb
 - kvredis.c, 74
- lspg_array2ptrs
 - lspg.c, 102
 - pgpmac.h, 278
- lspg_cmd_cb
 - lspg.c, 103
- lspg_connectPoll_response
 - kvredis.c, 84
 - lspg.c, 132
- lspg_demandairrights
 - lspg.c, 132
 - pgpmac.h, 308
- lspg_demandairrights_all
 - lspg.c, 103
 - pgpmac.h, 279
- lspg_demandairrights_call
 - lspg.c, 104
- lspg_demandairrights_cb
 - lspg.c, 104
- lspg_demandairrights_init
 - lspg.c, 104
- lspg_demandairrights_struct, 20
 - cond, 20
 - mutex, 20
 - new_value_ready, 20
- lspg_demandairrights_t
 - pgpmac.h, 274
- lspg_demandairrights_wait
 - lspg.c, 104
- lspg_flush
 - kvredis.c, 75
 - lspg.c, 104
- lspg_getcenter
 - lspg.c, 132
 - pgpmac.h, 309
- lspg_getcenter_all
 - lspg.c, 105
- lspg_getcenter_call
 - lspg.c, 105
 - pgpmac.h, 280
- lspg_getcenter_cb
 - lspg.c, 105
- lspg_getcenter_done
 - lspg.c, 106
 - pgpmac.h, 280
- lspg_getcenter_init
 - lspg.c, 106
- lspg_getcenter_struct, 20
 - cond, 21
 - dax, 21
 - dax_isnull, 21
 - day, 21
 - day_isnull, 22
 - daz, 22
 - daz_isnull, 22
 - dcx, 22
 - dcx_isnull, 22
 - dcy, 22
 - dcy_isnull, 22

- mutex, 22
- new_value_ready, 22
- no_rows_returned, 22
- zoom, 23
- zoom_isnull, 23
- lspg_getcenter_t
 - pgpmac.h, 274
- lspg_getcenter_wait
 - lspg.c, 107
 - pgpmac.h, 280
- lspg_getcurrentsampleid
 - lspg.c, 132
 - pgpmac.h, 309
- lspg_getcurrentsampleid_call
 - lspg.c, 107
- lspg_getcurrentsampleid_cb
 - lspg.c, 107
- lspg_getcurrentsampleid_init
 - lspg.c, 107
- lspg_getcurrentsampleid_read
 - lspg.c, 108
- lspg_getcurrentsampleid_struct, 23
 - cond, 23
 - getcurrentsampleid, 23
 - getcurrentsampleid_isnull, 24
 - mutex, 24
 - new_value_ready, 24
 - no_rows_returned, 24
- lspg_getcurrentsampleid_t
 - pgpmac.h, 274
- lspg_getcurrentsampleid_wait_for_id
 - lspg.c, 108
 - pgpmac.h, 280
- lspg_init
 - lspg.c, 108
 - pgpmac.h, 280
- lspg_lock_detector
 - lspg.c, 132
- lspg_lock_detector_all
 - lspg.c, 109
- lspg_lock_detector_call
 - lspg.c, 109
- lspg_lock_detector_cb
 - lspg.c, 109
- lspg_lock_detector_done
 - lspg.c, 109
- lspg_lock_detector_init
 - lspg.c, 109
- lspg_lock_detector_struct, 24
 - cond, 24
 - mutex, 24
 - new_value_ready, 25
- lspg_lock_detector_t
 - lspg.c, 101
- lspg_lock_detector_wait
 - lspg.c, 110
- lspg_lock_diffractionmeter
 - lspg.c, 133
- lspg_lock_diffractionmeter_all
 - lspg.c, 110
- lspg_lock_diffractionmeter_call
 - lspg.c, 110
- lspg_lock_diffractionmeter_cb
 - lspg.c, 110
- lspg_lock_diffractionmeter_done
 - lspg.c, 111
- lspg_lock_diffractionmeter_init
 - lspg.c, 111
- lspg_lock_diffractionmeter_struct, 25
 - cond, 25
 - mutex, 25
 - new_value_ready, 25
- lspg_lock_diffractionmeter_t
 - lspg.c, 101
- lspg_lock_diffractionmeter_wait
 - lspg.c, 111
- lspg_next_state
 - kvredis.c, 75
 - lspg.c, 111
- lspg_nextaction_cb
 - lspg.c, 112
- lspg_nextsample
 - lspg.c, 133
 - pgpmac.h, 309
- lspg_nextsample_all
 - lspg.c, 113
 - pgpmac.h, 281
- lspg_nextsample_call
 - lspg.c, 113
- lspg_nextsample_cb
 - lspg.c, 113
- lspg_nextsample_done
 - lspg.c, 114
- lspg_nextsample_init
 - lspg.c, 114
- lspg_nextsample_struct, 25
 - cond, 26
 - mutex, 26
 - new_value_ready, 26
 - nextsample, 26
 - nextsample_isnull, 26
 - no_rows_returned, 26
- lspg_nextsample_t
 - pgpmac.h, 274
- lspg_nextsample_wait
 - lspg.c, 114
- lspg_nextshot
 - lspg.c, 133
 - pgpmac.h, 309
- lspg_nextshot_call
 - lspg.c, 114
 - pgpmac.h, 281
- lspg_nextshot_cb
 - lspg.c, 115
- lspg_nextshot_done
 - lspg.c, 118

- pgpmac.h, 281
- lspg_nextshot_init
 - lspg.c, 119
- lspg_nextshot_struct, 27
 - active, 29
 - active2, 29
 - active2_isnull, 30
 - active_isnull, 30
 - ax, 30
 - ax2, 30
 - ax2_isnull, 30
 - ax_isnull, 30
 - ay, 30
 - ay2, 30
 - ay2_isnull, 30
 - ay_isnull, 30
 - az, 30
 - az2, 31
 - az2_isnull, 31
 - az_isnull, 31
 - cond, 31
 - cx, 31
 - cx2, 31
 - cx2_isnull, 31
 - cx_isnull, 31
 - cy, 31
 - cy2, 31
 - cy2_isnull, 32
 - cy_isnull, 32
 - dsdir, 32
 - dsdir_isnull, 32
 - dsdist, 32
 - dsdist2, 32
 - dsdist2_isnull, 32
 - dsdist_isnull, 32
 - dsexp, 32
 - dsexp2, 32
 - dsexp2_isnull, 33
 - dsexp_isnull, 33
 - dshpid, 33
 - dshpid_isnull, 33
 - dskappa, 33
 - dskappa2, 33
 - dskappa2_isnull, 33
 - dskappa_isnull, 33
 - dsnrg, 33
 - dsnrg2, 33
 - dsnrg2_isnull, 34
 - dsnrg_isnull, 34
 - dsomega, 34
 - dsomega2, 34
 - dsomega2_isnull, 34
 - dsomega_isnull, 34
 - dsoscaxis, 34
 - dsoscaxis2, 34
 - dsoscaxis2_isnull, 34
 - dsoscaxis_isnull, 34
 - dsowidth, 34
 - dsowidth2, 35
 - dsowidth2_isnull, 35
 - dsowidth_isnull, 35
 - dsphi, 35
 - dsphi2, 35
 - dsphi2_isnull, 35
 - dsphi_isnull, 35
 - dspid, 35
 - dspid_isnull, 35
 - mutex, 35
 - new_value_ready, 36
 - no_rows_returned, 36
 - sfn, 36
 - sfn_isnull, 36
 - sindex, 36
 - sindex2, 36
 - sindex2_isnull, 36
 - sindex_isnull, 36
 - skey, 36
 - skey_isnull, 36
 - sstart, 37
 - sstart2, 37
 - sstart2_isnull, 37
 - sstart_isnull, 37
 - stype, 37
 - stype2, 37
 - stype2_isnull, 37
 - stype_isnull, 37
- lspg_nextshot_t
 - pgpmac.h, 274
- lspg_nextshot_wait
 - lspg.c, 119
 - pgpmac.h, 282
- lspg_notice_processor
 - kvredis.c, 76
 - lspg.c, 119
- lspg_pg_connect
 - kvredis.c, 76
 - lspg.c, 119
- lspg_pg_service
 - kvredis.c, 77
 - lspg.c, 120
- lspg_query_next
 - kvredis.c, 78
 - lspg.c, 122
- lspg_query_push
 - kvredis.c, 79
 - lspg.c, 122
 - pgpmac.h, 282
- lspg_query_queue
 - kvredis.c, 84
 - lspg.c, 133
- lspg_query_queue_off
 - kvredis.c, 84
 - lspg.c, 133
- lspg_query_queue_on
 - kvredis.c, 84
 - lspg.c, 133

- lspg_query_queue_reply
 - kvredis.c, [84](#)
 - lspg.c, [133](#)
- lspg_query_queue_t
 - kvredis.c, [72](#)
 - pgpmac.h, [274](#)
- lspg_query_reply_next
 - kvredis.c, [79](#)
 - lspg.c, [123](#)
- lspg_query_reply_peek
 - kvredis.c, [80](#)
 - lspg.c, [123](#)
- lspg_queue_cond
 - lspg.c, [133](#)
- lspg_queue_mutex
 - lspg.c, [133](#)
- lspg_receive
 - kvredis.c, [80](#)
 - lspg.c, [123](#)
- lspg_resetPoll_response
 - kvredis.c, [84](#)
 - lspg.c, [134](#)
- lspg_run
 - lspg.c, [124](#)
 - pgpmac.h, [282](#)
- lspg_send_next_query
 - kvredis.c, [81](#)
 - lspg.c, [124](#)
- lspg_seq_run_prep
 - lspg.c, [134](#)
- lspg_seq_run_prep_all
 - lspg.c, [125](#)
 - pgpmac.h, [283](#)
- lspg_seq_run_prep_call
 - lspg.c, [125](#)
- lspg_seq_run_prep_cb
 - lspg.c, [126](#)
- lspg_seq_run_prep_done
 - lspg.c, [126](#)
- lspg_seq_run_prep_init
 - lspg.c, [126](#)
- lspg_seq_run_prep_struct, [37](#)
 - cond, [38](#)
 - mutex, [38](#)
 - new_value_ready, [38](#)
- lspg_seq_run_prep_t
 - lspg.c, [101](#)
- lspg_seq_run_prep_wait
 - lspg.c, [127](#)
- lspg_sig_service
 - lspg.c, [127](#)
- lspg_starttransfer
 - lspg.c, [134](#)
 - pgpmac.h, [309](#)
- lspg_starttransfer_all
 - lspg.c, [127](#)
- lspg_starttransfer_call
 - lspg.c, [128](#)
 - pgpmac.h, [283](#)
- lspg_starttransfer_cb
 - lspg.c, [128](#)
- lspg_starttransfer_done
 - lspg.c, [128](#)
 - pgpmac.h, [283](#)
- lspg_starttransfer_init
 - lspg.c, [128](#)
- lspg_starttransfer_struct, [38](#)
 - cond, [39](#)
 - mutex, [39](#)
 - new_value_ready, [39](#)
 - no_rows_returned, [39](#)
 - starttransfer, [39](#)
- lspg_starttransfer_t
 - pgpmac.h, [274](#)
- lspg_starttransfer_wait
 - lspg.c, [128](#)
 - pgpmac.h, [283](#)
- lspg_thread
 - lspg.c, [134](#)
- lspg_wait_for_detector
 - lspg.c, [134](#)
- lspg_wait_for_detector_all
 - lspg.c, [129](#)
- lspg_wait_for_detector_call
 - lspg.c, [129](#)
- lspg_wait_for_detector_cb
 - lspg.c, [129](#)
- lspg_wait_for_detector_done
 - lspg.c, [129](#)
- lspg_wait_for_detector_init
 - lspg.c, [130](#)
- lspg_wait_for_detector_struct, [39](#)
 - cond, [40](#)
 - mutex, [40](#)
 - new_value_ready, [40](#)
- lspg_wait_for_detector_t
 - lspg.c, [101](#)
- lspg_wait_for_detector_wait
 - lspg.c, [130](#)
- lspg_waitcryo
 - lspg.c, [134](#)
 - pgpmac.h, [309](#)
- lspg_waitcryo_all
 - lspg.c, [130](#)
 - pgpmac.h, [284](#)
- lspg_waitcryo_cb
 - lspg.c, [130](#)
 - pgpmac.h, [284](#)
- lspg_waitcryo_init
 - lspg.c, [130](#)
- lspg_waitcryo_struct, [40](#)
 - cond, [40](#)
 - mutex, [40](#)
 - new_value_ready, [41](#)
- lspg_waitcryo_t
 - pgpmac.h, [274](#)

lspg_worker
 lspg.c, 131
lspg_zoom_lut_call
 pgpmac.h, 284
lspgQueryQueueStruct, 41
 onResponse, 41
 qs, 41
lspgfd
 kvredis.c, 84
 lspg.c, 134
lspmac.c, 135
 _lspmac_motor_init, 147
 alignx, 201
 aligny, 201
 alignz, 201
 anal, 201
 apery, 201
 aperz, 201
 arm_parked, 201
 blight, 201
 blight_down, 201
 blight_f, 201
 blight_ud, 202
 blight_up, 202
 copy, 202
 capz, 202
 cenx, 202
 ceny, 202
 cleanstr, 148
 cr_cmd, 202
 cryo, 202
 cryo_back, 202
 cryo_switch, 203
 dbmem, 203
 dbmemIn, 203
 dryer, 203
 etel_init_ok, 203
 etel_on, 203
 etel_ready, 203
 ethCmdOff, 203
 ethCmdOn, 203
 ethCmdQueue, 204
 ethCmdReply, 204
 flight, 204
 flight_f, 204
 flight_oo, 204
 fluo, 204
 fluor_back, 204
 fscint, 204
 fshut, 204
 gb_cmd, 205
 getivars, 205
 getmvars, 205
 hex_dump, 148
 hp_air, 205
 kappa, 205
 LS_PMAC_STATE_CR, 143
 LS_PMAC_STATE_GB, 144
 LS_PMAC_STATE_GMR, 144
 LS_PMAC_STATE_IDLE, 144
 LS_PMAC_STATE_RR, 144
 LS_PMAC_STATE_SC, 144
 LS_PMAC_STATE_WACK, 144
 LS_PMAC_STATE_WCR, 144
 LS_PMAC_STATE_WGB, 144
 LSPMAC_PRESET_REGEX, 145
 lp_air, 205
 ls_pmac_state, 205
 lsConnect, 149
 lspmac_Error, 154
 lspmac_GetAllVars, 161
 lspmac_GetAllVarsCB, 162
 lspmac_GetAllMVars, 162
 lspmac_GetAllMVarsCB, 162
 lspmac_GetShortReplyCB, 164
 lspmac_Getmem, 163
 lspmac_GetmemReplyCB, 163
 lspmac_Reset, 187
 lspmac_SendControlReplyPrintCB, 192
 lspmac_Service, 192
 lspmac_SockFlush, 195
 lspmac_SockGetmem, 195
 lspmac_SockSendControlCharPrint, 196
 lspmac_SockSendDPControlChar, 196
 lspmac_SockSendDPControlCharCB, 196
 lspmac_SockSendDPLine, 196
 lspmac_SockSendDPQueue, 197
 lspmac_SockSendline, 197
 lspmac_SockSendline_nr, 198
 lspmac_abort, 150
 lspmac_ascii_buffers, 205
 lspmac_ascii_buffers_mutex, 205
 lspmac_ascii_buffers_t, 147
 lspmac_ascii_busy, 205
 lspmac_ascii_mutex, 206
 lspmac_asciicmdCB, 150
 lspmac_backLight_down_cb, 150
 lspmac_backLight_up_cb, 150
 lspmac_bi_init, 150
 lspmac_bis, 206
 lspmac_blight_lut_setup, 151
 lspmac_bo_init, 151
 lspmac_bo_read, 152
 lspmac_command_done_cb, 152
 lspmac_cryoSwitchChanged_cb, 153
 lspmac_dac_init, 153
 lspmac_dac_read, 153
 lspmac_dpascii_off, 206
 lspmac_dpascii_on, 206
 lspmac_dpascii_queue, 206
 lspmac_dpascii_queue_t, 147
 lspmac_est_move_time, 154
 lspmac_flight_lut_setup, 155
 lspmac_fscint_lut_setup, 156
 lspmac_fshut_init, 156
 lspmac_get_ascii, 157

- lspmac_get_ascii_cb, 157
- lspmac_get_status, 158
- lspmac_get_status_cb, 158
- lspmac_getBIPosition, 163
- lspmac_getPosition, 163
- lspmac_home1_queue, 164
- lspmac_home2_queue, 165
- lspmac_init, 166
- lspmac_jogabs_queue, 169
- lspmac_light_zoom_cb, 170
- lspmac_lut, 170
- lspmac_more_ascii_cb, 171
- lspmac_motor_init, 171
- lspmac_motors, 206
- lspmac_move_or_jog_abs_queue, 172
- lspmac_move_or_jog_preset_queue, 174
- lspmac_move_preset_queue, 175
- lspmac_moveabs_blight_factor_queue, 175
- lspmac_moveabs_bo_queue, 175
- lspmac_moveabs_flight_factor_queue, 176
- lspmac_moveabs_frontlight_oo_queue, 176
- lspmac_moveabs_fshut_queue, 177
- lspmac_moveabs_queue, 177
- lspmac_moveabs_timed_queue, 177
- lspmac_moveabs_wait, 178
- lspmac_movedac_queue, 180
- lspmac_movezoom_queue, 180
- lspmac_moving_cond, 206
- lspmac_moving_flags, 206
- lspmac_moving_mutex, 206
- lspmac_nbis, 206
- lspmac_next_state, 181
- lspmac_nmotors, 207
- lspmac_pmacmotor_read, 182
- lspmac_pop_queue, 185
- lspmac_pop_reply, 186
- lspmac_push_queue, 186
- lspmac_reset_queue, 187
- lspmac_rlut, 187
- lspmac_run, 188
- lspmac_scint_dried_cb, 189
- lspmac_scint_inPosition_cb, 190
- lspmac_send_command, 190
- lspmac_sendcmd, 191
- lspmac_sendcmd_nocb, 191
- lspmac_shutter_cond, 207
- lspmac_shutter_has_opened, 207
- lspmac_shutter_mutex, 207
- lspmac_shutter_read, 194
- lspmac_shutter_state, 207
- lspmac_soft_motor_init, 198
- lspmac_soft_motor_read, 198
- lspmac_status_last_time, 207
- lspmac_status_time, 207
- lspmac_test_preset, 198
- lspmac_video_rotate, 199
- lspmac_worker, 199
- lspmac_zoom_lut_setup, 200
- md2_status, 207
- md2_status_mutex, 207
- md2_status_t, 147
- minikappa_ok, 208
- now, 208
- omega, 208
- omega_zero_search, 208
- omega_zero_time, 208
- omega_zero_velocity, 208
- PMAC_MIN_CMD_TIME, 145
- PMACPORT, 145
- phi, 208
- pmac_cmd_size, 145
- pmac_error_strs, 208
- pmac_queue_cond, 209
- pmac_queue_mutex, 209
- pmac_thread, 209
- pmacfd, 209
- rr_cmd, 209
- sample_detected, 209
- scint, 209
- shutter_open, 210
- smart_mag_err, 210
- smart_mag_off, 210
- smart_mag_on, 210
- smart_mag_oo, 210
- VR_CTRL_RESPONSE, 145
- VR_DOWNLOAD, 145
- VR_FWDOWNLOAD, 145
- VR_IPADDRESS, 145
- VR_PMAC_FLUSH, 145
- VR_PMAC_GETBUFFER, 146
- VR_PMAC_GETLINE, 146
- VR_PMAC_GETMEM, 146
- VR_PMAC_GETRESPONSE, 146
- VR_PMAC_PORT, 146
- VR_PMAC_READREADY, 146
- VR_PMAC_SENDBIT, 146
- VR_PMAC_SETBIT, 146
- VR_PMAC_SETBITS, 146
- VR_PMAC_SETMEM, 146
- VR_PMAC_WRITEBUFFER, 146
- VR_PMAC_WRITEERROR, 147
- VR_UPLOAD, 147
- zoom, 210
- lspmac_Error
 - lspmac.c, 154
- lspmac_GetAllIVars
 - lspmac.c, 161
- lspmac_GetAllIVarsCB
 - lspmac.c, 162
- lspmac_GetAllMVars
 - lspmac.c, 162
- lspmac_GetAllMVarsCB
 - lspmac.c, 162
- lspmac_GetShortReplyCB
 - lspmac.c, 164
- lspmac_Getmem

- [lspmac.c](#), [163](#)
- [lspmac_GetmemReplyCB](#)
 - [lspmac.c](#), [163](#)
- [lspmac_Reset](#)
 - [lspmac.c](#), [187](#)
- [lspmac_SendControlReplyPrintCB](#)
 - [lspmac.c](#), [192](#)
- [lspmac_Service](#)
 - [lspmac.c](#), [192](#)
- [lspmac_SockFlush](#)
 - [lspmac.c](#), [195](#)
- [lspmac_SockGetmem](#)
 - [lspmac.c](#), [195](#)
- [lspmac_SockSendControlCharPrint](#)
 - [lspmac.c](#), [196](#)
- [lspmac_SockSendDPControlChar](#)
 - [lspmac.c](#), [196](#)
- [lspmac_SockSendDPControlCharCB](#)
 - [lspmac.c](#), [196](#)
- [lspmac_SockSendDPline](#)
 - [lspmac.c](#), [196](#)
 - [pgpmac.h](#), [295](#)
- [lspmac_SockSendDPqueue](#)
 - [lspmac.c](#), [197](#)
- [lspmac_SockSendline](#)
 - [lspmac.c](#), [197](#)
 - [pgpmac.h](#), [296](#)
- [lspmac_SockSendline_nr](#)
 - [lspmac.c](#), [198](#)
- [lspmac_abort](#)
 - [lspmac.c](#), [150](#)
 - [pgpmac.h](#), [284](#)
- [lspmac_ascii_buffers](#)
 - [lspmac.c](#), [205](#)
- [lspmac_ascii_buffers_mutex](#)
 - [lspmac.c](#), [205](#)
- [lspmac_ascii_buffers_struct](#), [42](#)
 - [command_buf](#), [42](#)
 - [command_buf_cc](#), [42](#)
 - [command_str](#), [42](#)
 - [response_buf](#), [42](#)
 - [response_n](#), [42](#)
 - [response_str](#), [42](#)
- [lspmac_ascii_buffers_t](#)
 - [lspmac.c](#), [147](#)
- [lspmac_ascii_busy](#)
 - [lspmac.c](#), [205](#)
- [lspmac_ascii_mutex](#)
 - [lspmac.c](#), [206](#)
- [lspmac_asciicmdCB](#)
 - [lspmac.c](#), [150](#)
- [lspmac_backLight_down_cb](#)
 - [lspmac.c](#), [150](#)
- [lspmac_backLight_up_cb](#)
 - [lspmac.c](#), [150](#)
- [lspmac_bi_init](#)
 - [lspmac.c](#), [150](#)
- [lspmac_bi_struct](#), [42](#)
 - [changeEventOff](#), [43](#)
 - [changeEventOn](#), [43](#)
 - [first_time](#), [43](#)
 - [mask](#), [43](#)
 - [mutex](#), [43](#)
 - [position](#), [44](#)
 - [previous](#), [44](#)
 - [ptr](#), [44](#)
- [lspmac_bi_t](#)
 - [pgpmac.h](#), [274](#)
- [lspmac_bis](#)
 - [lspmac.c](#), [206](#)
- [lspmac_blight_lut_setup](#)
 - [lspmac.c](#), [151](#)
- [lspmac_bo_init](#)
 - [lspmac.c](#), [151](#)
- [lspmac_bo_read](#)
 - [lspmac.c](#), [152](#)
- [lspmac_cmd_queue_struct](#), [44](#)
 - [event](#), [45](#)
 - [no_reply](#), [45](#)
 - [onResponse](#), [45](#)
 - [pcmd](#), [45](#)
 - [time_sent](#), [45](#)
- [lspmac_command_done_cb](#)
 - [lspmac.c](#), [152](#)
- [lspmac_cryoSwitchChanged_cb](#)
 - [lspmac.c](#), [153](#)
- [lspmac_dac_init](#)
 - [lspmac.c](#), [153](#)
- [lspmac_dac_read](#)
 - [lspmac.c](#), [153](#)
- [lspmac_dpascii_off](#)
 - [lspmac.c](#), [206](#)
- [lspmac_dpascii_on](#)
 - [lspmac.c](#), [206](#)
- [lspmac_dpascii_queue](#)
 - [lspmac.c](#), [206](#)
- [lspmac_dpascii_queue_struct](#), [45](#)
 - [event](#), [45](#)
 - [pl](#), [45](#)
- [lspmac_dpascii_queue_t](#)
 - [lspmac.c](#), [147](#)
- [lspmac_est_move_time](#)
 - [lspmac.c](#), [154](#)
- [lspmac_flight_lut_setup](#)
 - [lspmac.c](#), [155](#)
- [lspmac_fscint_lut_setup](#)
 - [lspmac.c](#), [156](#)
- [lspmac_fshut_init](#)
 - [lspmac.c](#), [156](#)
- [lspmac_get_ascii](#)
 - [lspmac.c](#), [157](#)
- [lspmac_get_ascii_cb](#)
 - [lspmac.c](#), [157](#)
- [lspmac_get_status](#)
 - [lspmac.c](#), [158](#)
- [lspmac_get_status_cb](#)

- lspmac.c, 158
- lspmac_getBIPosition
 - lspmac.c, 163
 - pgpmac.h, 284
- lspmac_getPosition
 - lspmac.c, 163
 - pgpmac.h, 285
- lspmac_home1_queue
 - lspmac.c, 164
 - pgpmac.h, 285
- lspmac_home2_queue
 - lspmac.c, 165
- lspmac_init
 - lspmac.c, 166
 - pgpmac.h, 286
- lspmac_jogabs_queue
 - lspmac.c, 169
 - pgpmac.h, 289
- lspmac_light_zoom_cb
 - lspmac.c, 170
- lspmac_lut
 - lspmac.c, 170
- lspmac_more_ascii_cb
 - lspmac.c, 171
- lspmac_motor_init
 - lspmac.c, 171
- lspmac_motor_struct, 46
 - active, 48
 - active_init, 48
 - actual_pos_cnts, 48
 - actual_pos_cnts_p, 48
 - axis, 48
 - command_sent, 48
 - cond, 49
 - coord_num, 49
 - dac_mvar, 49
 - home, 49
 - homing, 49
 - inactive_init, 49
 - lut, 49
 - max_accel, 49
 - max_pos, 49
 - max_speed, 50
 - min_pos, 50
 - motion_seen, 50
 - motor_num, 50
 - moveAbs, 50
 - mutex, 50
 - name, 50
 - neg_limit_hit, 50
 - neutral_pos, 50
 - nlut, 51
 - not_done, 51
 - pos_limit_hit, 51
 - position, 51
 - pq, 51
 - precision, 51
 - printf_fmt, 51
 - read, 51
 - read_mask, 51
 - read_ptr, 52
 - redis_fmt, 52
 - redis_position, 52
 - reported_position, 52
 - requested_pos_cnts, 52
 - requested_position, 52
 - status1, 52
 - status1_p, 52
 - status2, 52
 - status2_p, 53
 - status_str, 53
 - u2c, 53
 - unit, 53
 - update_resolution, 53
 - win, 53
 - write_fmt, 53
- lspmac_motor_t
 - pgpmac.h, 274
- lspmac_motors
 - lspmac.c, 206
 - pgpmac.h, 309
- lspmac_move_or_jog_abs_queue
 - lspmac.c, 172
 - pgpmac.h, 289
- lspmac_move_or_jog_preset_queue
 - lspmac.c, 174
 - pgpmac.h, 292
- lspmac_move_or_jog_queue
 - pgpmac.h, 292
- lspmac_move_preset_queue
 - lspmac.c, 175
 - pgpmac.h, 292
- lspmac_moveabs_blight_factor_queue
 - lspmac.c, 175
- lspmac_moveabs_bo_queue
 - lspmac.c, 175
- lspmac_moveabs_flight_factor_queue
 - lspmac.c, 176
- lspmac_moveabs_frontlight_oo_queue
 - lspmac.c, 176
- lspmac_moveabs_fshut_queue
 - lspmac.c, 177
- lspmac_moveabs_queue
 - lspmac.c, 177
 - pgpmac.h, 293
- lspmac_moveabs_timed_queue
 - lspmac.c, 177
- lspmac_moveabs_wait
 - lspmac.c, 178
 - pgpmac.h, 293
- lspmac_movedac_queue
 - lspmac.c, 180
- lspmac_movezoom_queue
 - lspmac.c, 180
- lspmac_moving_cond
 - lspmac.c, 206

- pgpmac.h, [309](#)
- lspmac_moving_flags
 - lspmac.c, [206](#)
 - pgpmac.h, [309](#)
- lspmac_moving_mutex
 - lspmac.c, [206](#)
 - pgpmac.h, [310](#)
- lspmac_nbis
 - lspmac.c, [206](#)
- lspmac_next_state
 - lspmac.c, [181](#)
- lspmac_nmotors
 - lspmac.c, [207](#)
 - pgpmac.h, [310](#)
- lspmac_pmacmotor_read
 - lspmac.c, [182](#)
- lspmac_pop_queue
 - lspmac.c, [185](#)
- lspmac_pop_reply
 - lspmac.c, [186](#)
- lspmac_push_queue
 - lspmac.c, [186](#)
- lspmac_reset_queue
 - lspmac.c, [187](#)
- lspmac_rlut
 - lspmac.c, [187](#)
- lspmac_run
 - lspmac.c, [188](#)
 - pgpmac.h, [294](#)
- lspmac_sample_detector_cb
 - lspg.c, [132](#)
- lspmac_scint_dried_cb
 - lspmac.c, [189](#)
- lspmac_scint_inPosition_cb
 - lspmac.c, [190](#)
- lspmac_send_command
 - lspmac.c, [190](#)
- lspmac_sendcmd
 - lspmac.c, [191](#)
- lspmac_sendcmd_nocb
 - lspmac.c, [191](#)
- lspmac_shutter_cond
 - lspmac.c, [207](#)
 - pgpmac.h, [310](#)
- lspmac_shutter_has_opened
 - lspmac.c, [207](#)
 - pgpmac.h, [310](#)
- lspmac_shutter_mutex
 - lspmac.c, [207](#)
 - pgpmac.h, [310](#)
- lspmac_shutter_read
 - lspmac.c, [194](#)
- lspmac_shutter_state
 - lspmac.c, [207](#)
 - pgpmac.h, [310](#)
- lspmac_soft_motor_init
 - lspmac.c, [198](#)
- lspmac_soft_motor_read
 - lspmac.c, [198](#)
- lspmac_status_last_time
 - lspmac.c, [207](#)
- lspmac_status_time
 - lspmac.c, [207](#)
- lspmac_test_preset
 - lspmac.c, [198](#)
- lspmac_video_rotate
 - lspmac.c, [199](#)
 - pgpmac.h, [296](#)
- lspmac_worker
 - lspmac.c, [199](#)
- lspmac_zoom_lut_setup
 - lspmac.c, [200](#)
- lsredis.c, [210](#)
 - _lsredis_get_obj, [213](#)
 - _lsredis_set_value, [214](#)
 - lsredis_addRead, [215](#)
 - lsredis_addWrite, [215](#)
 - lsredis_cleanup, [215](#)
 - lsredis_cmpnstr, [216](#)
 - lsredis_cmpstr, [216](#)
 - lsredis_cond, [227](#)
 - lsredis_debugCB, [216](#)
 - lsredis_delRead, [217](#)
 - lsredis_delWrite, [217](#)
 - lsredis_fd_service, [217](#)
 - lsredis_find_preset, [218](#)
 - lsredis_get_obj, [218](#)
 - lsredis_get_string_array, [219](#)
 - lsredis_getb, [219](#)
 - lsredis_getc, [219](#)
 - lsredis_getd, [220](#)
 - lsredis_getl, [220](#)
 - lsredis_getstr, [220](#)
 - lsredis_head, [227](#)
 - lsredis_hgetCB, [220](#)
 - lsredis_hstab, [227](#)
 - lsredis_init, [221](#)
 - lsredis_key_select_regex, [228](#)
 - lsredis_keysCB, [222](#)
 - lsredis_maybe_add_key, [222](#)
 - lsredis_mutex, [228](#)
 - lsredis_objs, [228](#)
 - lsredis_publisher, [228](#)
 - lsredis_regexec, [223](#)
 - lsredis_run, [223](#)
 - lsredis_running, [228](#)
 - lsredis_set_value, [223](#)
 - lsredis_setstr, [223](#)
 - lsredis_sig_service, [224](#)
 - lsredis_subCB, [225](#)
 - lsredis_thread, [228](#)
 - lsredis_worker, [226](#)
 - redisDisconnectCB, [227](#)
 - roac, [228](#)
 - rofd, [228](#)
 - subac, [228](#)

- subfd, 228
- wrac, 228
- wrfd, 228
- lsredis_addRead
 - lsredis.c, 215
- lsredis_addWrite
 - lsredis.c, 215
- lsredis_cleanup
 - lsredis.c, 215
- lsredis_cmpnstr
 - lsredis.c, 216
 - pgpmac.h, 297
- lsredis_cmpstr
 - lsredis.c, 216
 - pgpmac.h, 297
- lsredis_cond
 - lsredis.c, 227
- lsredis_debugCB
 - lsredis.c, 216
- lsredis_delRead
 - lsredis.c, 217
- lsredis_delWrite
 - lsredis.c, 217
- lsredis_fd_service
 - lsredis.c, 217
- lsredis_find_preset
 - lsredis.c, 218
 - pgpmac.h, 297
- lsredis_get_obj
 - lsredis.c, 218
 - pgpmac.h, 298
- lsredis_get_string_array
 - lsredis.c, 219
 - pgpmac.h, 298
- lsredis_getb
 - lsredis.c, 219
 - pgpmac.h, 299
- lsredis_getc
 - lsredis.c, 219
- lsredis_getd
 - lsredis.c, 220
 - pgpmac.h, 299
- lsredis_getl
 - lsredis.c, 220
 - pgpmac.h, 299
- lsredis_getstr
 - lsredis.c, 220
 - pgpmac.h, 299
- lsredis_head
 - lsredis.c, 227
- lsredis_hgetCB
 - lsredis.c, 220
- lsredis_htab
 - lsredis.c, 227
- lsredis_init
 - lsredis.c, 221
 - pgpmac.h, 300
- lsredis_key_select_regex
 - lsredis.c, 228
- lsredis_keysCB
 - lsredis.c, 222
- lsredis_maybe_add_key
 - lsredis.c, 222
- lsredis_mutex
 - lsredis.c, 228
- lsredis_obj_struct, 53
 - avalue, 54
 - bvalue, 54
 - cond, 54
 - cvalue, 55
 - dvalue, 55
 - events_name, 55
 - hits, 55
 - key, 55
 - lvalue, 55
 - mutex, 55
 - next, 55
 - valid, 55
 - value, 56
 - value_length, 56
 - wait_for_me, 56
- lsredis_obj_t
 - pgpmac.h, 274
- lsredis_objs
 - lsredis.c, 228
- lsredis_publisher
 - lsredis.c, 228
- lsredis_regex
 - lsredis.c, 223
 - pgpmac.h, 301
- lsredis_run
 - lsredis.c, 223
 - pgpmac.h, 301
- lsredis_running
 - lsredis.c, 228
- lsredis_set_value
 - lsredis.c, 223
- lsredis_setstr
 - lsredis.c, 223
 - pgpmac.h, 301
- lsredis_sig_service
 - lsredis.c, 224
- lsredis_subCB
 - lsredis.c, 225
- lsredis_thread
 - lsredis.c, 228
- lsredis_worker
 - lsredis.c, 226
- lstimer.c, 229
 - handler, 230
 - LSTIMER_LIST_LENGTH, 230
 - lstimer_active_timers, 234
 - lstimer_add_timer, 231
 - lstimer_cond, 234
 - lstimer_init, 231
 - lstimer_list, 234

- lstimer_list_t, 230
- lstimer_mutex, 234
- lstimer_run, 232
- lstimer_thread, 234
- lstimer_timerid, 235
- lstimer_worker, 232
- new_timer, 235
- service_timers, 233
- lstimer_active_timers
 - lstimer.c, 234
- lstimer_add_timer
 - lstimer.c, 231
 - pgpmac.h, 302
- lstimer_cond
 - lstimer.c, 234
- lstimer_init
 - lstimer.c, 231
 - pgpmac.h, 303
- lstimer_list
 - lstimer.c, 234
- lstimer_list_struct, 56
 - delay_nsecs, 57
 - delay_secs, 57
 - event, 57
 - init_nsecs, 57
 - init_secs, 57
 - last_nsecs, 57
 - last_secs, 57
 - ncalls, 57
 - next_nsecs, 58
 - next_secs, 58
 - shots, 58
- lstimer_list_t
 - lstimer.c, 230
- lstimer_mutex
 - lstimer.c, 234
- lstimer_run
 - lstimer.c, 232
 - pgpmac.h, 303
- lstimer_thread
 - lstimer.c, 234
- lstimer_timerid
 - lstimer.c, 235
- lstimer_worker
 - lstimer.c, 232
- lsupdate_init
 - pgpmac.h, 303
- lsupdate_run
 - pgpmac.h, 304
- ltime
 - lslogging_queue_struct, 19
- lut
 - lspmac_motor_struct, 49
- lvalue
 - lsredis_obj_struct, 55
- MD2CMDSD_CMD_LENGTH
 - pgpmac.h, 273
- main
 - kvredis.c, 81
 - pgpmac.c, 261
- mask
 - lspmac_bi_struct, 43
- max_accel
 - lspmac_motor_struct, 49
- max_pos
 - lspmac_motor_struct, 49
- max_speed
 - lspmac_motor_struct, 50
- md2_status
 - lspmac.c, 207
- md2_status_mutex
 - lspmac.c, 207
 - pgpmac.h, 310
- md2_status_t
 - lspmac.c, 147
- md2StatusStruct, 58
 - acc11c_1, 60
 - acc11c_2, 60
 - acc11c_3, 60
 - acc11c_5, 60
 - acc11c_6, 60
 - alignx_act_pos, 60
 - alignx_status_1, 60
 - alignx_status_2, 60
 - aligny_act_pos, 60
 - aligny_status_1, 60
 - aligny_status_2, 60
 - alignz_act_pos, 60
 - alignz_status_1, 60
 - alignz_status_2, 61
 - analyzer_act_pos, 61
 - analyzer_status_1, 61
 - analyzer_status_2, 61
 - aperturey_act_pos, 61
 - aperturey_status_1, 61
 - aperturey_status_2, 61
 - aperturez_act_pos, 61
 - aperturez_status_1, 61
 - aperturez_status_2, 61
 - back_dac, 61
 - copy_act_pos, 61
 - copy_status_1, 62
 - copy_status_2, 62
 - capz_act_pos, 62
 - capz_status_1, 62
 - capz_status_2, 62
 - centerx_act_pos, 62
 - centerx_status_1, 62
 - centerx_status_2, 62
 - centery_act_pos, 62
 - centery_status_1, 62
 - centery_status_2, 62
 - dummy1, 62
 - dummy2, 63
 - dummy3, 63
 - dummy4, 63

- dummy5, [63](#)
- dummy6, [63](#)
- dummy7, [63](#)
- dummy8, [63](#)
- dummy9, [63](#)
- dummyA, [63](#)
- dummyB, [63](#)
- front_dac, [63](#)
- fs_has_opened, [63](#)
- fs_has_opened_globally, [64](#)
- fs_is_open, [64](#)
- kappa_act_pos, [64](#)
- kappa_status_1, [64](#)
- kappa_status_2, [64](#)
- moving_flags, [64](#)
- number_passes, [64](#)
- omega_act_pos, [64](#)
- omega_status_1, [64](#)
- omega_status_2, [64](#)
- phi_act_pos, [64](#)
- phi_status_1, [64](#)
- phi_status_2, [65](#)
- phiscan, [65](#)
- scint_act_pos, [65](#)
- scint_piezo, [65](#)
- scint_status_1, [65](#)
- scint_status_2, [65](#)
- zoom_act_pos, [65](#)
- zoom_status_1, [65](#)
- zoom_status_2, [65](#)
- md2cmds.c, [235](#)
 - md2cmds_abort, [237](#)
 - md2cmds_action_queue, [237](#)
 - md2cmds_action_wait, [238](#)
 - md2cmds_capz_moving_time, [258](#)
 - md2cmds_center, [238](#)
 - md2cmds_cmd, [258](#)
 - md2cmds_collect, [238](#)
 - md2cmds_cond, [258](#)
 - md2cmds_coordsys_1_stopped_cb, [242](#)
 - md2cmds_coordsys_2_stopped_cb, [242](#)
 - md2cmds_coordsys_3_stopped_cb, [242](#)
 - md2cmds_coordsys_4_stopped_cb, [242](#)
 - md2cmds_coordsys_5_stopped_cb, [242](#)
 - md2cmds_coordsys_7_stopped_cb, [243](#)
 - md2cmds_home_prep, [243](#)
 - md2cmds_home_wait, [243](#)
 - md2cmds_homing_cond, [258](#)
 - md2cmds_homing_count, [259](#)
 - md2cmds_homing_mutex, [259](#)
 - md2cmds_init, [244](#)
 - md2cmds_is_moving, [244](#)
 - md2cmds_kappaphi_move, [244](#)
 - md2cmds_maybe_done_homing_cb, [244](#)
 - md2cmds_maybe_done_moving_cb, [245](#)
 - md2cmds_maybe_rotate_done_cb, [245](#)
 - md2cmds_md_status_code, [259](#)
 - md2cmds_move_prep, [245](#)
 - md2cmds_move_wait, [246](#)
 - md2cmds_moveAbs, [246](#)
 - md2cmds_moving_cond, [259](#)
 - md2cmds_moving_count, [259](#)
 - md2cmds_moving_mutex, [259](#)
 - md2cmds_moving_queue_wait, [259](#)
 - md2cmds_mutex, [259](#)
 - md2cmds_mvcenter_move, [247](#)
 - md2cmds_organs_move_presets, [248](#)
 - md2cmds_phase_change, [249](#)
 - md2cmds_prep_axis, [251](#)
 - md2cmds_rotate, [252](#)
 - md2cmds_rotate_cb, [253](#)
 - md2cmds_run, [254](#)
 - md2cmds_set_scale_cb, [254](#)
 - md2cmds_thread, [259](#)
 - md2cmds_time_capz_cb, [255](#)
 - md2cmds_transfer, [255](#)
 - md2cmds_worker, [258](#)
 - rotating, [259](#)
- md2cmds_abort
 - md2cmds.c, [237](#)
- md2cmds_action_queue
 - md2cmds.c, [237](#)
- md2cmds_action_wait
 - md2cmds.c, [238](#)
- md2cmds_capz_moving_time
 - md2cmds.c, [258](#)
- md2cmds_center
 - md2cmds.c, [238](#)
- md2cmds_cmd
 - md2cmds.c, [258](#)
 - pgpmac.h, [310](#)
- md2cmds_collect
 - md2cmds.c, [238](#)
- md2cmds_cond
 - md2cmds.c, [258](#)
 - pgpmac.h, [310](#)
- md2cmds_coordsys_1_stopped_cb
 - md2cmds.c, [242](#)
- md2cmds_coordsys_2_stopped_cb
 - md2cmds.c, [242](#)
- md2cmds_coordsys_3_stopped_cb
 - md2cmds.c, [242](#)
- md2cmds_coordsys_4_stopped_cb
 - md2cmds.c, [242](#)
- md2cmds_coordsys_5_stopped_cb
 - md2cmds.c, [242](#)
- md2cmds_coordsys_7_stopped_cb
 - md2cmds.c, [243](#)
- md2cmds_home_prep
 - md2cmds.c, [243](#)
- md2cmds_home_wait
 - md2cmds.c, [243](#)
- md2cmds_homing_cond
 - md2cmds.c, [258](#)
- md2cmds_homing_count
 - md2cmds.c, [259](#)

- md2cmds_homing_mutex
 - md2cmds.c, [259](#)
- md2cmds_init
 - md2cmds.c, [244](#)
 - pgpmac.h, [304](#)
- md2cmds_is_moving
 - md2cmds.c, [244](#)
- md2cmds_kappaphi_move
 - md2cmds.c, [244](#)
- md2cmds_maybe_done_homing_cb
 - md2cmds.c, [244](#)
- md2cmds_maybe_done_moving_cb
 - md2cmds.c, [245](#)
- md2cmds_maybe_rotate_done_cb
 - md2cmds.c, [245](#)
- md2cmds_md_status_code
 - md2cmds.c, [259](#)
 - pgpmac.h, [311](#)
- md2cmds_move_prep
 - md2cmds.c, [245](#)
- md2cmds_move_wait
 - md2cmds.c, [246](#)
- md2cmds_moveAbs
 - md2cmds.c, [246](#)
- md2cmds_moving_cond
 - md2cmds.c, [259](#)
- md2cmds_moving_count
 - md2cmds.c, [259](#)
- md2cmds_moving_mutex
 - md2cmds.c, [259](#)
- md2cmds_moving_queue_wait
 - md2cmds.c, [259](#)
- md2cmds_mutex
 - md2cmds.c, [259](#)
 - pgpmac.h, [311](#)
- md2cmds_mvcenter_move
 - md2cmds.c, [247](#)
- md2cmds_organs_move_presets
 - md2cmds.c, [248](#)
- md2cmds_pg_cond
 - pgpmac.h, [311](#)
- md2cmds_pg_mutex
 - pgpmac.h, [311](#)
- md2cmds_phase_change
 - md2cmds.c, [249](#)
- md2cmds_prep_axis
 - md2cmds.c, [251](#)
- md2cmds_rotate
 - md2cmds.c, [252](#)
- md2cmds_rotate_cb
 - md2cmds.c, [253](#)
- md2cmds_run
 - md2cmds.c, [254](#)
 - pgpmac.h, [304](#)
- md2cmds_set_scale_cb
 - md2cmds.c, [254](#)
- md2cmds_thread
 - md2cmds.c, [259](#)
- md2cmds_time_capz_cb
 - md2cmds.c, [255](#)
- md2cmds_transfer
 - md2cmds.c, [255](#)
- md2cmds_worker
 - md2cmds.c, [258](#)
- min_pos
 - lspmac_motor_struct, [50](#)
- minikappa_ok
 - lspmac.c, [208](#)
 - pgpmac.h, [311](#)
- mk_pgpmac_redis, [11](#)
 - b, [12](#)
 - bi_list, [12](#)
 - f, [12](#)
 - fnc, [12](#)
 - hard_ini, [12](#)
 - hard_ini_fields, [12](#)
 - head, [12](#)
 - hi, [12](#)
 - i, [12](#)
 - motor_dict, [13](#)
 - motor_field_lists, [13](#)
 - motor_presets, [13](#)
 - p, [13](#)
 - pi, [13](#)
 - ppos, [13](#)
 - pref_ini, [13](#)
 - v, [13](#)
 - x, [13](#)
 - y, [13](#)
 - zoom_settings, [13](#)
- mk_pgpmac_redis.py, [260](#)
- motion_seen
 - lspmac_motor_struct, [50](#)
- motor_dict
 - mk_pgpmac_redis, [13](#)
- motor_field_lists
 - mk_pgpmac_redis, [13](#)
- motor_num
 - lspmac_motor_struct, [50](#)
- motor_presets
 - mk_pgpmac_redis, [13](#)
- moveAbs
 - lspmac_motor_struct, [50](#)
- moving_flags
 - md2StatusStruct, [64](#)
- mutex
 - lspg_demandairrights_struct, [20](#)
 - lspg_getcenter_struct, [22](#)
 - lspg_getcurrentsampleid_struct, [24](#)
 - lspg_lock_detector_struct, [24](#)
 - lspg_lock_diffractionmeter_struct, [25](#)
 - lspg_nextsample_struct, [26](#)
 - lspg_nextshot_struct, [35](#)
 - lspg_seq_run_prep_struct, [38](#)
 - lspg_starttransfer_struct, [39](#)
 - lspg_wait_for_detector_struct, [40](#)

- lspg_waitcryo_struct, 40
- lspmac_bi_struct, 43
- lspmac_motor_struct, 50
- lsredis_obj_struct, 55
- name
 - lspmac_motor_struct, 50
- ncalls
 - lstimer_list_struct, 57
- ncurses_mutex
 - pgpmac.c, 264
 - pgpmac.h, 311
- neg_limit_hit
 - lspmac_motor_struct, 50
- neutral_pos
 - lspmac_motor_struct, 50
- new_timer
 - lstimer.c, 235
- new_value_ready
 - lspg_demandairrights_struct, 20
 - lspg_getcenter_struct, 22
 - lspg_getcurrentsampleid_struct, 24
 - lspg_lock_detector_struct, 25
 - lspg_lock_diffractionmeter_struct, 25
 - lspg_nextsample_struct, 26
 - lspg_nextshot_struct, 36
 - lspg_seq_run_prep_struct, 38
 - lspg_starttransfer_struct, 39
 - lspg_wait_for_detector_struct, 40
 - lspg_waitcryo_struct, 41
- next
 - lsevents_listener_struct, 18
 - lsredis_obj_struct, 55
- next_nsecs
 - lstimer_list_struct, 58
- next_secs
 - lstimer_list_struct, 58
- nextsample
 - lspg_nextsample_struct, 26
- nextsample_isnull
 - lspg_nextsample_struct, 26
- nlut
 - lspmac_motor_struct, 51
- no_reply
 - lspmac_cmd_queue_struct, 45
- no_rows_returned
 - lspg_getcenter_struct, 22
 - lspg_getcurrentsampleid_struct, 24
 - lspg_nextsample_struct, 26
 - lspg_nextshot_struct, 36
 - lspg_starttransfer_struct, 39
- not_done
 - lspmac_motor_struct, 51
- now
 - kvredis.c, 84
 - lspg.c, 134
 - lspmac.c, 208
- number_passes
 - md2StatusStruct, 64
- omega
 - lspmac.c, 208
 - pgpmac.h, 311
- omega_act_pos
 - md2StatusStruct, 64
- omega_status_1
 - md2StatusStruct, 64
- omega_status_2
 - md2StatusStruct, 64
- omega_zero_search
 - lspmac.c, 208
- omega_zero_time
 - lspmac.c, 208
 - pgpmac.h, 311
- omega_zero_velocity
 - lspmac.c, 208
- onResponse
 - lspgQueryQueueStruct, 41
 - lspmac_cmd_queue_struct, 45
- options
 - iniParser::iniParser, 16
- p
 - mk_pgpmac_redis, 13
- PMAC_MIN_CMD_TIME
 - lspmac.c, 145
- PMACPORT
 - lspmac.c, 145
- pcmd
 - lspmac_cmd_queue_struct, 45
- pgpmac.c, 260
 - main, 261
 - ncurses_mutex, 264
 - pgpmac_printf, 263
 - running, 264
 - stdinService, 263
 - stdinfda, 265
 - term_input, 265
 - term_output, 265
 - term_status, 265
 - term_status2, 265
- pgpmac.h, 265
 - _GNU_SOURCE, 273
 - alignx, 305
 - aligny, 305
 - alignz, 305
 - anal, 305
 - apery, 305
 - aperz, 305
 - arm_parked, 305
 - blight, 306
 - blight_down, 306
 - blight_f, 306
 - blight_ud, 306
 - blight_up, 306
 - copy, 306
 - capz, 306
 - cenx, 306
 - ceny, 306

cryo, 307
cryo_back, 307
cryo_switch, 307
dryer, 307
etel_init_ok, 307
etel_on, 307
etel_ready, 307
flight, 307
flight_f, 307
flight_oo, 308
fluo, 308
fluor_back, 308
fscint, 308
fshut, 308
hp_air, 308
kappa, 308
lp_air, 308
lsevents_add_listener, 275
lsevents_init, 275
lsevents_remove_listener, 276
lsevents_run, 276
lsevents_send_event, 276
lslogging_init, 277
lslogging_log_message, 277
lslogging_run, 278
lspg_array2ptrs, 278
lspg_demandairrights, 308
lspg_demandairrights_all, 279
lspg_demandairrights_t, 274
lspg_getcenter, 309
lspg_getcenter_call, 280
lspg_getcenter_done, 280
lspg_getcenter_t, 274
lspg_getcenter_wait, 280
lspg_getcurrentsampleid, 309
lspg_getcurrentsampleid_t, 274
lspg_getcurrentsampleid_wait_for_id, 280
lspg_init, 280
lspg_nextsample, 309
lspg_nextsample_all, 281
lspg_nextsample_t, 274
lspg_nextshot, 309
lspg_nextshot_call, 281
lspg_nextshot_done, 281
lspg_nextshot_t, 274
lspg_nextshot_wait, 282
lspg_query_push, 282
lspg_query_queue_t, 274
lspg_run, 282
lspg_seq_run_prep_all, 283
lspg_starttransfer, 309
lspg_starttransfer_call, 283
lspg_starttransfer_done, 283
lspg_starttransfer_t, 274
lspg_starttransfer_wait, 283
lspg_waitcryo, 309
lspg_waitcryo_all, 284
lspg_waitcryo_cb, 284
lspg_waitcryo_t, 274
lspg_zoom_lut_call, 284
lspmac_SockSendDPLine, 295
lspmac_SockSendline, 296
lspmac_abort, 284
lspmac_bi_t, 274
lspmac_getBIPosition, 284
lspmac_getPosition, 285
lspmac_home1_queue, 285
lspmac_init, 286
lspmac_jogabs_queue, 289
lspmac_motor_t, 274
lspmac_motors, 309
lspmac_move_or_jog_abs_queue, 289
lspmac_move_or_jog_preset_queue, 292
lspmac_move_or_jog_queue, 292
lspmac_move_preset_queue, 292
lspmac_moveabs_queue, 293
lspmac_moveabs_wait, 293
lspmac_moving_cond, 309
lspmac_moving_flags, 309
lspmac_moving_mutex, 310
lspmac_nmotors, 310
lspmac_run, 294
lspmac_shutter_cond, 310
lspmac_shutter_has_opened, 310
lspmac_shutter_mutex, 310
lspmac_shutter_state, 310
lspmac_video_rotate, 296
lsredis_cmpnstr, 297
lsredis_cmpstr, 297
lsredis_find_preset, 297
lsredis_get_obj, 298
lsredis_get_string_array, 298
lsredis_getb, 299
lsredis_getd, 299
lsredis_getl, 299
lsredis_getstr, 299
lsredis_init, 300
lsredis_obj_t, 274
lsredis_regexec, 301
lsredis_run, 301
lsredis_setstr, 301
lstimer_add_timer, 302
lstimer_init, 303
lstimer_run, 303
lsupdate_init, 303
lsupdate_run, 304
MD2CMDS_CMD_LENGTH, 273
md2_status_mutex, 310
md2cmds_cmd, 310
md2cmds_cond, 310
md2cmds_init, 304
md2cmds_md_status_code, 311
md2cmds_mutex, 311
md2cmds_pg_cond, 311
md2cmds_pg_mutex, 311
md2cmds_run, 304

- minikappa_ok, 311
- ncurses_mutex, 311
- omega, 311
- omega_zero_time, 311
- pgpmac_printf, 304
- phi, 311
- pmac_cmd_queue_t, 274
- pmac_cmd_t, 275
- pmac_queue_cond, 311
- pmac_queue_mutex, 311
- PmacSockSendline, 305
- sample_detected, 312
- scint, 312
- shutter_open, 312
- smart_mag_err, 312
- smart_mag_off, 312
- smart_mag_on, 312
- smart_mag_oo, 312
- term_input, 312
- term_output, 312
- term_status, 313
- term_status2, 313
- zoom, 313
- pgpmac_printf
 - pgpmac.c, 263
 - pgpmac.h, 304
- phi
 - lspmac.c, 208
 - pgpmac.h, 311
- phi_act_pos
 - md2StatusStruct, 64
- phi_status_1
 - md2StatusStruct, 64
- phi_status_2
 - md2StatusStruct, 65
- phiscan
 - md2StatusStruct, 65
- pi
 - mk_pgpmac_redis, 13
- pl
 - lspmac_dpascii_queue_struct, 45
- pmac_cmd_queue_t
 - pgpmac.h, 274
- pmac_cmd_size
 - lspmac.c, 145
- pmac_cmd_t
 - pgpmac.h, 275
- pmac_error_strs
 - lspmac.c, 208
- pmac_queue_cond
 - lspmac.c, 209
 - pgpmac.h, 311
- pmac_queue_mutex
 - lspmac.c, 209
 - pgpmac.h, 311
- pmac_thread
 - lspmac.c, 209
- PmacSockSendline
 - pgpmac.h, 305
- pmacfd
 - lspmac.c, 209
- pos_limit_hit
 - lspmac_motor_struct, 51
- position
 - lspmac_bi_struct, 44
 - lspmac_motor_struct, 51
- ppos
 - mk_pgpmac_redis, 13
- pq
 - lspmac_motor_struct, 51
- precision
 - lspmac_motor_struct, 51
- pref_ini
 - mk_pgpmac_redis, 13
- previous
 - lspmac_bi_struct, 44
- printf_fmt
 - lspmac_motor_struct, 51
- ptr
 - lspmac_bi_struct, 44
- q
 - kvredis.c, 84
 - lspg.c, 134
- qs
 - lspgQueryQueueStruct, 41
- raw_regexp
 - lsevents_listener_struct, 18
- re
 - lsevents_listener_struct, 18
- read
 - iniParser::iniParser, 16
 - lspmac_motor_struct, 51
- read_mask
 - lspmac_motor_struct, 51
- read_ptr
 - lspmac_motor_struct, 52
- redis_fmt
 - lspmac_motor_struct, 52
- redis_position
 - lspmac_motor_struct, 52
- redisDisconnectCB
 - kvredis.c, 83
 - lsredis.c, 227
- reported_position
 - lspmac_motor_struct, 52
- Request
 - tagEthernetCmd, 66
- RequestType
 - tagEthernetCmd, 66
- requested_pos_cnts
 - lspmac_motor_struct, 52
- requested_position
 - lspmac_motor_struct, 52
- response_buf
 - lspmac_ascii_buffers_struct, 42

- response_n
 - lspmac_ascii_buffers_struct, 42
- response_str
 - lspmac_ascii_buffers_struct, 42
- roac
 - lsredis.c, 228
- rofd
 - lsredis.c, 228
- rotating
 - md2cmds.c, 259
- rr_cmd
 - lspmac.c, 209
- running
 - pgpmac.c, 264
- sample_detected
 - lspmac.c, 209
 - pgpmac.h, 312
- scint
 - lspmac.c, 209
 - pgpmac.h, 312
- scint_act_pos
 - md2StatusStruct, 65
- scint_piezo
 - md2StatusStruct, 65
- scint_status_1
 - md2StatusStruct, 65
- scint_status_2
 - md2StatusStruct, 65
- sd
 - iniParser::iniParser, 17
- sections
 - iniParser::iniParser, 17
- service_timers
 - lstimer.c, 233
- sfn
 - lspg_nextshot_struct, 36
- sfn_isnull
 - lspg_nextshot_struct, 36
- shots
 - lstimer_list_struct, 58
- shutter_open
 - lspmac.c, 210
 - pgpmac.h, 312
- sindex
 - lspg_nextshot_struct, 36
- sindex2
 - lspg_nextshot_struct, 36
- sindex2_isnull
 - lspg_nextshot_struct, 36
- sindex_isnull
 - lspg_nextshot_struct, 36
- skey
 - lspg_nextshot_struct, 36
- skey_isnull
 - lspg_nextshot_struct, 36
- smart_mag_err
 - lspmac.c, 210
 - pgpmac.h, 312
- smart_mag_off
 - lspmac.c, 210
 - pgpmac.h, 312
- smart_mag_on
 - lspmac.c, 210
 - pgpmac.h, 312
- smart_mag_oo
 - lspmac.c, 210
 - pgpmac.h, 312
- sstart
 - lspg_nextshot_struct, 37
- sstart2
 - lspg_nextshot_struct, 37
- sstart2_isnull
 - lspg_nextshot_struct, 37
- sstart_isnull
 - lspg_nextshot_struct, 37
- starttransfer
 - lspg_starttransfer_struct, 39
- status1
 - lspmac_motor_struct, 52
- status1_p
 - lspmac_motor_struct, 52
- status2
 - lspmac_motor_struct, 52
- status2_p
 - lspmac_motor_struct, 53
- status_str
 - lspmac_motor_struct, 53
- stdinService
 - pgpmac.c, 263
- stdinfda
 - pgpmac.c, 265
- stype
 - lspg_nextshot_struct, 37
- stype2
 - lspg_nextshot_struct, 37
- stype2_isnull
 - lspg_nextshot_struct, 37
- stype_isnull
 - lspg_nextshot_struct, 37
- subac
 - kvredis.c, 85
 - lsredis.c, 228
- subfd
 - kvredis.c, 85
 - lsredis.c, 228
- tagEthernetCmd, 65
 - bData, 66
 - Request, 66
 - RequestType, 66
 - wIndex, 66
 - wLength, 66
 - wValue, 66
- term_input
 - pgpmac.c, 265
 - pgpmac.h, 312
- term_output

- pgpmac.c, [265](#)
- pgpmac.h, [312](#)
- term_status
 - pgpmac.c, [265](#)
 - pgpmac.h, [313](#)
- term_status2
 - pgpmac.c, [265](#)
 - pgpmac.h, [313](#)
- time_sent
 - lspmac_cmd_queue_struct, [45](#)
- u2c
 - lspmac_motor_struct, [53](#)
- unit
 - lspmac_motor_struct, [53](#)
- update_resolution
 - lspmac_motor_struct, [53](#)
- v
 - mk_pgpmac_redis, [13](#)
- VR_CTRL_RESPONSE
 - lspmac.c, [145](#)
- VR_DOWNLOAD
 - lspmac.c, [145](#)
- VR_FWDOWNLOAD
 - lspmac.c, [145](#)
- VR_IPADDRESS
 - lspmac.c, [145](#)
- VR_PMAC_FLUSH
 - lspmac.c, [145](#)
- VR_PMAC_GETBUFFER
 - lspmac.c, [146](#)
- VR_PMAC_GETLINE
 - lspmac.c, [146](#)
- VR_PMAC_GETMEM
 - lspmac.c, [146](#)
- VR_PMAC_GETRESPONSE
 - lspmac.c, [146](#)
- VR_PMAC_PORT
 - lspmac.c, [146](#)
- VR_PMAC_READREADY
 - lspmac.c, [146](#)
- VR_PMAC_SENDCTRLCHAR
 - lspmac.c, [146](#)
- VR_PMAC_SENDLINE
 - lspmac.c, [146](#)
- VR_PMAC_SETBIT
 - lspmac.c, [146](#)
- VR_PMAC_SETBITS
 - lspmac.c, [146](#)
- VR_PMAC_SETMEM
 - lspmac.c, [146](#)
- VR_PMAC_WRITEBUFFER
 - lspmac.c, [146](#)
- VR_PMAC_WRITEERROR
 - lspmac.c, [147](#)
- VR_UPLOAD
 - lspmac.c, [147](#)
- valid
 - lsredis_obj_struct, [55](#)
- value
 - lsredis_obj_struct, [56](#)
- value_length
 - lsredis_obj_struct, [56](#)
- wIndex
 - tagEthernetCmd, [66](#)
- wLength
 - tagEthernetCmd, [66](#)
- wValue
 - tagEthernetCmd, [66](#)
- wait_for_me
 - lsredis_obj_struct, [56](#)
- win
 - lspmac_motor_struct, [53](#)
- wrac
 - lsredis.c, [228](#)
- wrfd
 - lsredis.c, [228](#)
- write_fmt
 - lspmac_motor_struct, [53](#)
- x
 - mk_pgpmac_redis, [13](#)
- y
 - mk_pgpmac_redis, [13](#)
- zoom
 - lspg_getcenter_struct, [23](#)
 - lspmac.c, [210](#)
 - pgpmac.h, [313](#)
- zoom_act_pos
 - md2StatusStruct, [65](#)
- zoom_isnull
 - lspg_getcenter_struct, [23](#)
- zoom_settings
 - mk_pgpmac_redis, [13](#)
- zoom_status_1
 - md2StatusStruct, [65](#)
- zoom_status_2
 - md2StatusStruct, [65](#)